Intégration de données à partir de connaissances : une approche multi-agent pour la gestion des changements

Rahee Ghurbhurn

Ecole Nationale Supérieure des Mines
SAINT-ETIENNE

N° d'ordre : 480 I

THESE

présentée par

Rahee Ghurbhurn

Pour obtenir le grade de Docteur
de l'Ecole Nationale Supérieure des Mines de Saint-Etienne

Spécialité : Informatique

Intégration de données à partir de connaissances une approche multi-agent pour la gestion des changements

Soutenue à Saint Étienne le 27 mai 2008

Membres du jury

Président :		
	Abdelkader Djamel ZIGHED	Professeur, Université Lumière, Lyon
Rapporteurs :		
	Jérôme GENSEL	Professeur, Université Pierre Mendès-France, Grenoble
	HACID Mohand-Saïd	Professeur, Université Claude Bernard , Lyon
Examinateurs :		
	Pierre LEBRUN	Ingénieur-Responsable Archi. IT /STMicroelectronics
	Philippe BEAUNE	Maitre Assistant, ENS Mines Saint-Etienne
	Olivier BOISSIER	Professeur, ENS Mines Saint-Etienne
Directeur(s) de thèse :		
	Olivier BOISSIER	Professeur, ENS Mines Saint-Etienne

◘ Spécialités doctorales : **Responsables :**

SCIENCES ET GENIE DES MATERIAUX	**J. DRIVER** Directeur de recherche – Centre SMS
MECANIQUE ET INGENIERIE	**A. VAUTRIN** Professeur – Centre SMS
GENIE DES PROCEDES	**G. THOMAS** Professeur – Centre SPIN
SCIENCES DE LA TERRE	**B. GUY** Maitre de recherche – Centre SPIN
SCIENCES ET GENIE DE L'ENVIRONNEMENT	**J. BOURGOIS** Professeur – Centre SITE
MATHEMATIQUES APPLIQUEES	**E. TOUBOUL** Ingénieur – Centre G2I
INFORMATIQUE	**O. BOISSIER** Professeur – Centre G2I
IMAGE, VISION, SIGNAL	**JC. PINOLI** Professeur – Centre CIS
GENIE INDUSTRIEL	**P. BURLAT** Professeur – Centre G2I
MICROELECTRONIQUE	**Ph. COLLOT** Professeur – Centre CMP

◘ Enseignants-chercheurs et chercheurs autorisés à diriger des thèses de doctorat (titulaires d'un doctorat d'Etat ou d'une HDR)

AVRIL	Stéphane	MA	Mécanique & Ingénierie	CIS
BATTON-HUBERT	Mireille	MA	Sciences & Génie de l'Environnement	SITE
BENABEN	Patrick	PR 2	Sciences & Génie des Matériaux	SMS
BERNACHE-ASSOLANT	Didier	PR 1	Génie des Procédés	CIS
BIGOT	Jean-Pierre	MR	Génie des Procédés	SPIN
BILAL	Essaïd	DR	Sciences de la Terre	SPIN
BOISSIER	Olivier	PR 2	Informatique	G2I
BOUCHER	Xavier	MA	Génie Industriel	G2I
BOUDAREL	Marie-Reine	MA	Sciences de l'inform. & com.	DF
BOURGOIS	Jacques	PR 1	Sciences & Génie de l'Environnement	SITE
BRODHAG	Christian	MR	Sciences & Génie de l'Environnement	SITE
BURLAT	Patrick	PR 2	Génie industriel	G2I
CARRARO	Laurent	PR 1	Mathématiques Appliquées	G2I
COLLOT	Philippe	PR 1	Microélectronique	CMP
COURNIL	Michel	PR 1	Génie des Procédés	SPIN
DAUZERE-PERES	Stéphane	PR 1	Génie industriel	CMP
DARRIEULAT	Michel	ICM	Sciences & Génie des Matériaux	SMS
DECHOMETS	Roland	PR 1	Sciences & Génie de l'Environnement	SITE
DESRAYAUD	Christophe	MA	Mécanique & Ingénierie	SMS
DELAFOSSE	David	PR 2	Sciences & Génie des Matériaux	SMS
DOLGUI	Alexandre	PR 1	Génie Industriel	G2I
DRAPIER	Sylvain	PR 2	Mécanique & Ingénierie	CIS
DRIVER	Julian	DR	Sciences & Génie des Matériaux	SMS
FOREST	Bernard	PR 1	Sciences & Génie des Matériaux	CIS
FORMISYN	Pascal	PR 1	Sciences & Génie de l'Environnement	SITE
FORTUNIER	Roland	PR 1	Sciences & Génie des Matériaux	CMP
FRACZKIEWICZ	Anna	MR	Sciences & Génie des Matériaux	SMS
GARCIA	Daniel	CR	Génie des Procédés	SPIN
GIRARDOT	Jean-Jacques	MR	Informatique	G2I
GOEURIOT	Dominique	MR	Sciences & Génie des Matériaux	SMS
GOEURIOT	Patrice	MR	Sciences & Génie des Matériaux	SMS
GRAILLOT	Didier	DR	Sciences & Génie de l'Environnement	SITE
GROSSEAU	Philippe	MR	Génie des Procédés	SPIN
GRUY	Frédéric	MR	Génie des Procédés	SPIN
GUILHOT	Bernard	DR	Génie des Procédés	CIS
GUY	Bernard	MR	Sciences de la Terre	SPIN
GUYONNET	René	DR	Génie des Procédés	SPIN
HERRI	Jean-Michel	PR 2	Génie des Procédés	SPIN
KLÖCKER	Helmut	MR	Sciences & Génie des Matériaux	SMS
LAFOREST	Valérie	CR	Sciences & Génie de l'Environnement	SITE
LI	Jean-Michel	EC (CCI MP)	Microélectronique	CMP
LONDICHE	Henry	MR	Sciences & Génie de l'Environnement	SITE
MOLIMARD	Jérôme	MA	Sciences & Génie des Matériaux	SMS
MONTHEILLET	Frank	DR 1 CNRS	Sciences & Génie des Matériaux	SMS
PERIER-CAMBY	Laurent	PR1	Génie des Procédés	SPIN
PIJOLAT	Christophe	PR 1	Génie des Procédés	SPIN
PIJOLAT	Michèle	PR 1	Génie des Procédés	SPIN
PINOLI	Jean-Charles	PR 1	Image, Vision, Signal	CIS
STOLARZ	Jacques	CR	Sciences & Génie des Matériaux	SMS
SZAFNICKI	Konrad	CR	Sciences de la Terre	SITE
THOMAS	Gérard	PR 1	Génie des Procédés	SPIN
VALDIVIESO	François	MA	Sciences & Génie des Matériaux	SMS
VAUTRIN	Alain	PR 1	Mécanique & Ingénierie	SMS
VIRICELLE	Jean-Paul	MR	Génie des procédés	SPIN
WOLSKI	Krzysztof	CR	Sciences & Génie des Matériaux	SMS
XIE	Xiaolan	PR 1	Génie industriel	CIS

Glossaire :

PR 1 Professeur 1ère catégorie
PR 2 Professeur 2ème catégorie
MA(MDC) Maître assistant
DR (DR1) Directeur de recherche
Ing. Ingénieur
MR(DR2) Maître de recherche
CR Chargé de recherche
EC Enseignant-chercheur
ICM Ingénieur en chef des mines

Centres :

SMS Sciences des Matériaux et des Structures
SPIN Sciences des Processus Industriels et Naturels
SITE Sciences Information et Technologies pour l'Environnement
G2I Génie Industriel et Informatique
CMP Centre de Microélectronique de Provence
CIS Centre Ingénierie et Santé

A ma mère, à mon père.

Remerciements.

Je remercie les membres du jury, Monsieur Jérôme Gensel et Monsieur Mohand-Saïd Hacid d'avoir accepté de rapporter ce travail. Merci à Monsieur Djamel Zighed et à Monsieur Pierre Lebrun d'avoir accepté de faire partie du jury.

Je remercie particulièrement Philippe Beaune et Olivier Boissier de m'avoir accueilli au sein de l'équipe SMA, je les remercie également pour leur disponibilité et leurs critiques scientifiques. Je remercie mes collègues Julien, Ludivine, Guillaume, Laurent, Cosmin, Max pour nos échanges et nos discussions constructives ainsi que le personnel du département G2I, Liliane, Maryline, Jean-François, Steven.

Je tiens à remercier le Centre Microelectronics de Provence de m'avoir financé pendant ces trois années. Je remercie particulièrement Stéphane Dauzère-Péres et Véronique Villareal.

Merci à Pierre Lebrun, Hugues Solignac et Thomas Groussin pour leur accueil et leur aide chez STMicroelectonics.

Merci à mes amis du billard one, pour leur soutien et leur écoute.

Merci à Amande et JC pour les « succulents » Saumur Champigny. Merci à Sam et Isa pour les nombreuses soirées passées ensemble. Merci à vous quatre pour les picons et les moments de rigolade. Merci à Matthieu et Adeline pour nos sympathiques discussions, Stephanie merci de m'avoir écouté de temps en temps.

Merci à Cécile pour avoir été là, les chemins se croisent, se recroisent ; l'important est de savoir apprécier le voyage.

Merci à ma famille, ma mère et mon père que je ne remercierai jamais assez pour tout ce qu'ils ont fait. Merci à mes frères pour les moments passés ensemble.

J'en oublie surement, alors merci à tous ceux qui ont contribué de près ou de loin à la réalisation de ce travail.

Résumé

Au sein des entreprises, un composant vital dans la prise de décision, qu'elle soit globale ou locale, est le système d'information. Celui-ci contient, en effet, les informations nécessaires pour décider, agir, apprendre, comprendre, prévoir et contrôler. Sa structure est généralement liée à l'histoire de l'entreprise dans le sens où cet ensemble s'est constitué d'éléments qui se sont juxtaposés au fil du temps au gré des choix stratégiques formant in fine un ensemble complexe et hétérogène. L'existence de plusieurs systèmes d'information, au sein de grande sociétés, pouvant rendre la recherche d'information difficile pour les utilisateurs métiers. L'objectif de nos travaux consiste à permettre aux utilisateurs d'explorer les connaissances, formulées en terme métier, contenues dans plusieurs systèmes d'informations et de récupérer les données qui leur sont associées. Il s'agit donc de mettre à disposition des utilisateurs une vue globale des connaissances disponibles liées à leurs domaines métiers, tout en dissimulant la diversité des sources d'informations et en garantissant que les données associées sont récupérables malgré les changements qui peuvent se produire au sein des différents systèmes. Dans cette thèse nous proposons KRISMAS, un solution d'intégration de données basée une représentation des connaissances métiers et une architecture Multi-Agents. La représentation des connaissances prend la forme d'une ou plusieurs ontologies de domaine permettant aux utilisateurs d'explorer les connaissances des sources de données et de formuler des requêtes pour la récupération des données. Les agents sont utilisés pour l'exploitation de la représentation des connaissances métiers ainsi que pour la gestion des changements au sein d'un système d'intégration

MOTS-CLÉS : Intégration de données, Gestion de changements, Système Multi-Agents, Ontologie, Web Sémantique.

Abstract

Organisations are generally composed of several information systems which are key components in the decision making process. Finding specific functional information in these systems may prove to be difficult for non IT users. Data integration systems are used to centralise data from several information systems but the actual solutions simply expose the collected data and don't manage changes that may occur in the underlying data sources structure. The objective of our work is to express the collected data in terms of business knowledge and allow functional users to explore knowledge and retrieve the corresponding data without having to know the underlying data sources structures. These data are found in distributed and heterogeneous data sources belonging to different services or departments. Our proposal, KRISMAS, is based on ontologies modelling functional domains and a multi-agent architecture performing the data retrieval and managing the changes that may occur within the data sources. By managing the changes we mean maintaining the ontologies coherent with the data sources.

KEYWORDS: Data integration, Change management, Multi Agents System, Ontology, Information Retrieval.

Table des matières

Index des figures

Index des Définitions

Chapitre I.

Introduction

La problématique que nous abordons dans ce manuscrit est l'*intégration sémantique des données*. Nous l'avons étudié au sein du groupe STMicroelectronics dans le cadre de la stratégie d'*évolution* de ses systèmes d'information. Avant de présenter la thèse que nous défendons, nous allons préciser le contexte industriel de notre étude pour en cerner la problématique. Nous la préciserons ensuite d'un point de vue de son contexte scientifique.

I.1 Contexte industriel

La mondialisation économique engendre une quête vers une plus grande flexibilité et une plus grande capacité à s'adapter afin de faire face à une concurrence accrue. Les entreprises dans leur ensemble sont de moins en moins isolées et établissent de nouvelles collaborations, s'implantent sur de nouveaux marchés en lançant de nouveaux produits et/ou s'implantant dans de nouveaux pays. L'ensemble de ces orientations décidées au niveau stratégique ne sont pas sans conséquences dans la mesure où des services entiers peuvent être externalisés alors que d'autres deviennent synergiques grâce à des collaborations avec différents partenaires. Ces évolutions engendrent généralement des évolutions dans le cœur de métier des entreprises, le langage de communication entre les partenaires, ainsi que le besoin informationnel pour la prise de décision, évoluent.

Un composant vital dans la prise de décision, qu'elle soit globale ou locale, est le système

d'information. Celui-ci contient, en effet, les informations nécessaires pour décider, agir, apprendre, comprendre, prévoir et contrôler. Sa structure est généralement liée à l'histoire de l'entreprise dans le sens où cet ensemble s'est constitué d'éléments qui se sont juxtaposés au fil du temps au gré des choix stratégiques formant *in fine* un ensemble complexe et hétérogène. Les éléments qui le composent ont pour principal objectif le recueil, le traitement, le stockage et la diffusion d'informations. Suivant le niveau de décision, ces éléments peuvent être:

- le système d'information stratégique;
- le système d'information de gestion;
- le système d'information opérationnel;

Dans la réalité, il n'existe pas vraiment de distinction. Le système d'information intègre des ensembles d'applications et d'outils permettant de remplir les différentes fonctions des systèmes d'information stratégique, de gestion ou opérationnel.

Dans un contexte où les structures évoluent au gré de délocalisations, d'externalisations, de fusions ou de développement de partenariats, et où la flexibilité et la souplesse sont nécessaires pour survivre à la concurrence, les entreprises cherchent à développer leurs capacités de communication interne et externe afin de mieux agir et réagir. Il est possible de distinguer plusieurs niveaux de communication : la *communication interne*, la *communication externe vers les partenaires* et la *communication externe vers les clients*.

La communication interne prend la forme d'une plus grande efficacité de la communication entre applications. Ce type de communication est généralement appelé *Application to Application* (A2A).

La communication externe vers les partenaires consiste à créer un réseaux d'échange d'information pour optimiser la gestion de la chaîne logistique ou favoriser une plus grande synergie dans le cadre de partenariats. Ce type de communication est généralement appelé *Business to Business* (B2B)

La communication externe vers les clients (*business to consumer* B2C) consiste à extraire le maximum d'informations des données sur les clients afin de mieux comprendre leurs besoins, personnaliser les produits, définir la stratégie de prix mais aussi gérer les commandes clients, l'après vente et les enquêtes de satisfaction.

L'ensemble de ces fonctions reposent sur des applications, appartenant à différents services, qui sont généralement regroupées au sein de systèmes d'information afin de fournir une vision transversale aux utilisateurs et aux décideurs.

Obtenir une telle *vision transversale* est loin d'être trivial car cela nécessite l'identification des données pertinentes par rapport aux métiers de l'entreprise. En effet, les données dans l'entreprise sont caractérisées par la *redondance* et l'*inconsistance* qui engendrent une faible productivité et des sur-coûts en matière de traitements.

La redondance est principalement liée au fait que les différents services s'intéressent à différentes facettes d'une même chose (client, produit, équipement...). De ce fait, les informations sont saisies et re-saisies dans différents systèmes suivant différentes règles de gestion. Cette redondance implique naturellement des problèmes d'inconsistance dès qu'on souhaite avoir une vision transversale. Pour chaque services les données contenues dans ses systèmes sont justes et valables mais, consolidées, elles sont potentiellement conflictuelles. Baser des outils de A2A, de B2B ou de B2C sur de telles données engendre des coûts supplémentaires en matière de *réconciliation des données*, des difficultés dans la gestion de commandes clients pouvant aller jusqu'à les rendre insatisfaits des services rendus et la complexification de la gestion des processus métiers intra-entreprise ou inter-entreprise.

Ces dysfonctionnements liés à la redondance de données sont exacerbés par un contexte environnemental ponctué par des évolutions technologiques plus rapides, un time-to-market[1] de plus en plus court et des fusions-acquisitions plus fréquentes.

I.2 Contexte scientifique

L'intégration, au sens large, peut être mise en place, au sein d'une entreprise, à différents niveaux. Nous pouvons, en effet, distinguer le niveau des utilisateurs, le niveau des processus métiers, le niveau des applications, et le niveau des données.

L'*intégration au niveau des utilisateurs* permet de donner **accès aux données** de l'entreprise

1 Le délai nécessaire pour le développement et la mise au point d'un projet ou d'un produit, avant qu'il puisse être lancé sur le marché. Avec le raccourcissement des cycles de vie des produits, il est devenu un facteur stratégique majeur, en ce sens où une réduction caractérisée du time to market peut permettre à une entreprise et/ou une marque d'améliorer de manière significative sa rentabilité mais aussi lui donner la possibilité de prendre un avantage concurrentiel décisif.

ainsi qu'aux ressources du système d'information. Cette intégration prend généralement la forme de portails utilisateurs ou de portails mis à la disposition des partenaires. Les portails peuvent être déclinés en portail d'informations d'entreprise ou portail d'applications d'entreprise.

L'*intégration au niveau des processus* consiste à modéliser le fonctionnement réel de l'entreprise afin d'obtenir une vue globale des processus métiers et des interactions avec les différentes applications et les utilisateurs. Cette vue globale est utilisée pour l'optimisation des échanges ou pour faciliter l'**automatisation des processus métiers** à l'aide d'applications métiers.

L'*intégration au niveau des applications* consiste à optimiser les échanges inter-applicatif. Cette optimisation consiste à identifier, coordonner et gérer les flux échangés (transactions, messages ou données) entre les applications au sein d'une entreprise voire entre partenaires.

L'*intégration au niveau des données* a pour objectif de fournir une **vue unifiée des données** disséminées au sein d'une entreprise. Plusieurs technologies ont été proposées pour l'intégration de données permettant d'améliorer leur qualité et de réduire leur inconsistance. Les trois grandes familles sont :

- L'*Extract Transform and Load* (ETL) qui consiste à **extraire les données** d'une ou plusieurs sources de données, nettoyer et mettre en forme les données avant de les charger dans une source de données cible.
- L'*Enterprise Application Integration* (EAI) se positionne au niveau des applications. L'objectif de l'EAI consiste à **faire communiquer les applications** par des échanges de messages ou de données. La mise en place de ce type d'intégration nécessite l'utilisation d'un *middleware,* appelé Enterprise Service Bus (ESB), qui prend en charge la communication inter-applications. Les technologies utilisées pour la communication sont principalement les Web Services, les connecteurs JCA et XML.
- L'*Entreprise Information Integration* (EII) se positionne comme l'ETL au niveau des données. L'objectif de l'EII consiste à **présenter les informations** contenues dans différentes sources d'information comme si ces dernières étaient présentes dans une source unique. Associée à cette représentation homogène des données, l'EII permet aussi aux utilisateurs de récupérer les données, généralement grâce à la formulation de requêtes. L'EII est composée de solutions issues du domaine des bases de données mais aussi d'XML.

Ces trois familles de technologies ne sont *pas concurrentes mais complémentaires*. Elles se positionnent de la manière suivante. Comme le montre la Figure 1 l'ETL est une solution qui permet de définir des tâches d'intégration à des intervalles de temps réguliers alors que EII est une solution d'intégration plus orientée temps réel; l'intégration de données se fait au moment de la demande. L'EAI est, quant à elle, une solution orientée vers une intégration des processus au sein d'un entreprise.

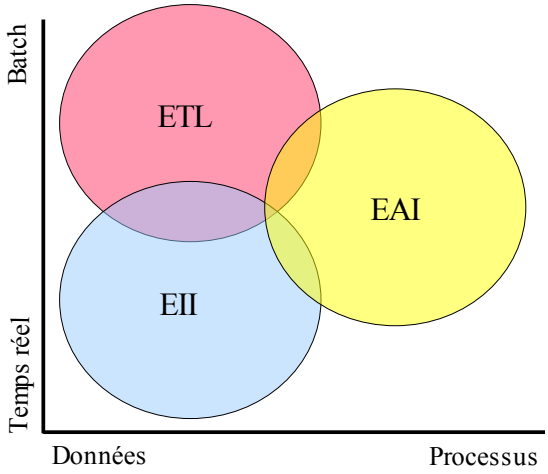

Figure 1: Positionnement ETL, EAI, EII

I.3 Thèse défendue

Nos travaux s'inscrivent dans le cadre de l'EII et l'objectif est d'enrichir cette approche à travers une **modélisation métier** des données et un mécanisme de **mise à jour des correspondances** entre la représentation métier et les sources de données.

Chapron[1] définit un objet métier comme " *un ensemble d'informations manipulé par des processus informationnels dont la signification unique est partagée par l'ensemble des acteurs quelque soit le processus. Les objets métiers sont donc des concepts trans-organisationnels dans l'entreprise. Ces objets métiers sont des entités qui sont principalement utilisés par des acteurs métiers, moins par des acteurs du système informatique qui sont habitués à manipuler des données beaucoup plus précises. Ces objets métiers correspondent*

aux concepts utilisés quotidiennement par les acteurs de l'entreprise pour communiquer entre eux et qu'ils utilisent naturellement lors de la description de leur processus."

Nous proposons donc de modéliser ces objets métiers à travers une **modélisation des connaissances métiers**. Cette représentation est construite par des spécialistes métiers et non par des spécialistes informatiques. Cette représentation permet aux utilisateurs d'avoir une **vision d'ensemble** des connaissances métiers présentes au sein de l'entreprise, ainsi que les interactions qui existent en elles.

Afin de permettre la récupération des données il est nécessaire d'établir un lien entre la représentation des connaissances métiers et les différentes sources de données. Cependant, du fait des changements organisationnels, technologiques ou concurrentiels ces liens sont amenés à évoluer dans le temps. Les liens évoluent soit parce que la connaissance modélisée évolue, soit parce que les sources de données évoluent. Nous souhaitons mettre en place un mécanisme de *détection de changements* au niveau des sources de données afin de mettre à jour automatiquement les liens avec la représentation des connaissances.

La mise en œuvre de ces propositions se base sur l'utilisation des technologies issues du web sémantique et de la technologie multi-agent. Le résultat de la mise en œuvre est un outil de visualisation des connaissances et de récupération des données disséminées au niveau des différents systèmes d'information.

I.4 Organisation du manuscrit

Ce document est organisé en trois parties. La première est un état de l'art sur les composants d'un système d'intégration de données (SID). Dans le *chapitre II*, nous donnons une **définition** de ce qu'est un système d'intégration et comment il peut être structuré. La Figure 2 illustre les composants du système d'intégration à savoir un **schéma global**, des **correspondances** et des sources de données. Ces composants peuvent être organisés sous la forme d'une architecture **médiateur** ou **pair-à-pair** (Peer Data Management System)

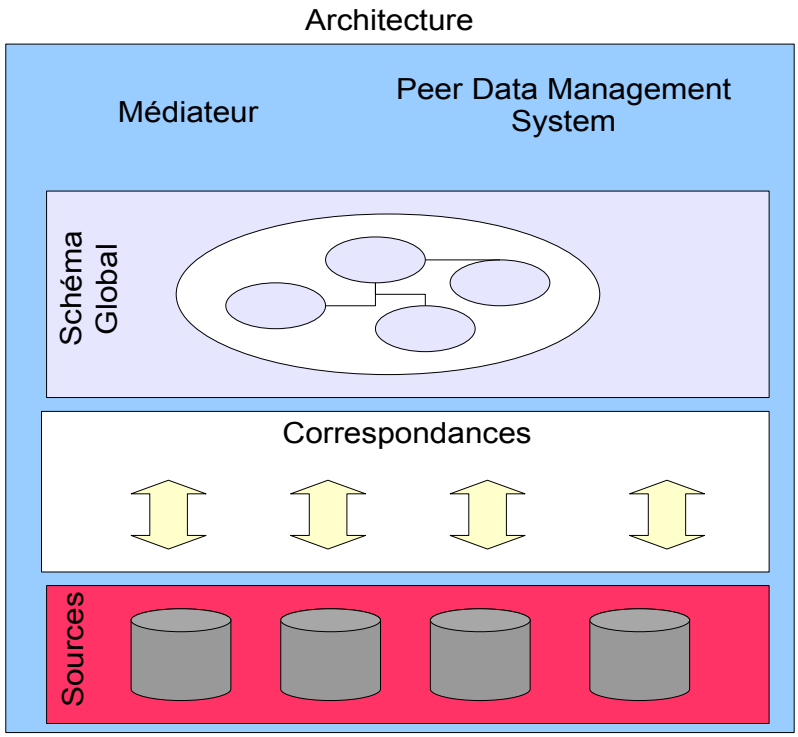

Figure 2: Composants et organisation d'un SID

Le *chapitre III* traite des différentes méthodes utilisées pour la **définition d'un schéma global**. Ces méthodes sont le modèle relationnel, le modèle objet, le méta-langage XML, et les logiques de description. Une fois le schéma global défini, il est possible de le relier aux sources de données. Le *chapitre IV* décrit les différents **types de correspondances** proposées dans la littérature. Les correspondances peuvent prendre la forme du **global-as-view**, l**ocal-as-view**, g**lobal local-as-view** et le **both-as-view**. Le principal inconvénient des approches existantes est la gestion des changements au niveau du schéma global et des sources de données. La Figure 3 illustre les solutions proposées pour chaque composant.

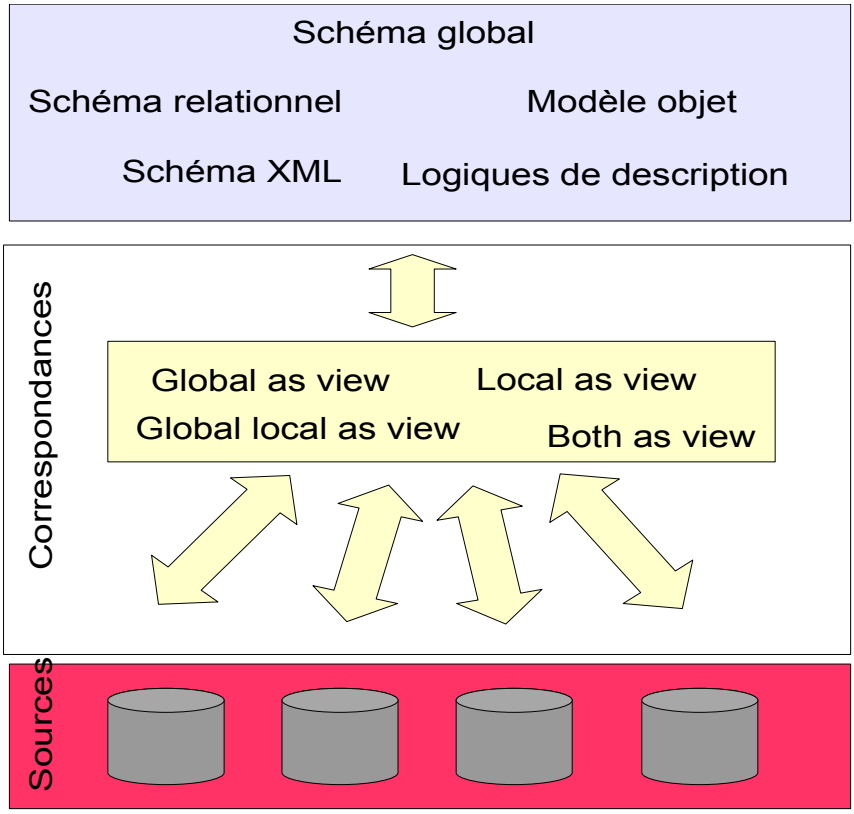

Figure 3: Solution existantes par composant.

Le *chapitre V* fait un tour d'horizon des différentes implémentations existantes en matière d'intégration. Ces solutions sont analysées selon trois critères: la *prise en compte de l'utilisateur* à travers la présentation sémantique des données, la *gestion du changement* au niveau du schéma global et des sources de données et la *récupération des données*.

Cette analyse nous conduit à proposer **KRISMAS** (Knowledge Representation Integration System with Multi-Agent System) afin d'améliorer les solutions existantes. Nos propositions sont détaillées dans la partie 2. Le *chapitre VI* détaille le **modèle de connaissances métiers** ainsi que les correspondances que nous proposons. La modélisation des connaissances métiers est la première étape dans la mise en place de notre proposition. Elle est définie avec l'aide des utilisateurs du système afin de mieux refléter le fonctionnement réel. Ces connaissances définies, nous procédons à l'identification de leur emplacement au sein des sources de données participant au SID et définissons des correspondances. Ces correspondances **ne sont pas définies comme des vues ou des règles de transformation.** Elles sont, par conséquent, plus flexibles et permettent de mieux gérer l'impact des changements au niveau du système d'intégration. Les connaissances métiers et les conrrespondances sont organisés au sein d'une

architecture basée sur les **systèmes multi-agents**. L'architecture que nous proposons permet d'abstraire le contenu des différentes sources de données et d'isoler les utilisateurs des modifications qu'elles peuvent subir. Les *chapitres VII et* VIII détaillent le **fonctionnement ainsi que les fonctionnalités du système**. La Figure 4 illustre les différentes briques de notre proposition.

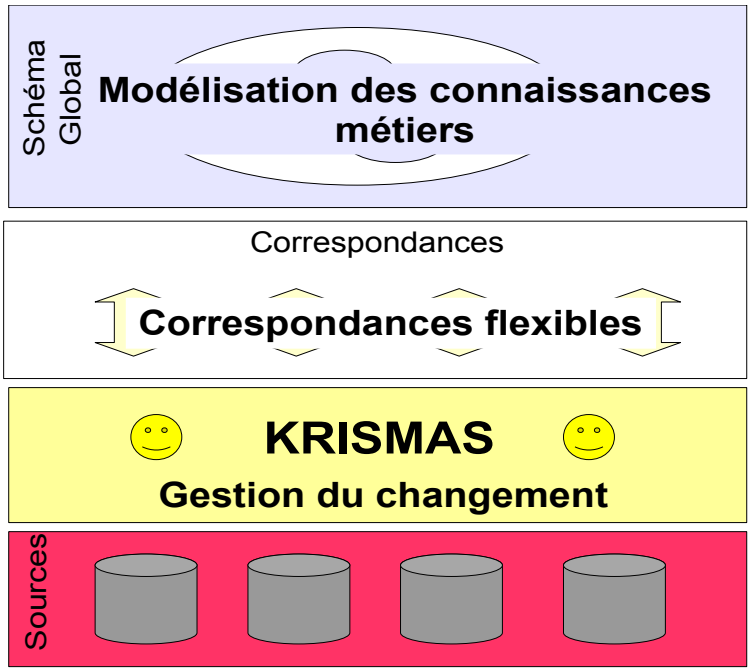

Figure 4: Propositions

Le *chapitre IX* de cette thèse concerne l'**implémentation** de nos propositions. Nous y détaillons les outils utilisés ainsi que les **protocoles** utilisés pour la mise en œuvre de nos propositions.

Le *chapitre X* conclu ce manuscrit en résumant nos propositions et en présentant quelques perspectives.

Partie I

État de l'art

Nous nous intéressons dans cette partie à l'étude de la manière dont est construite la représentation métier des données ainsi qu'à la manière dont est prise en compte la gestion des changements. Dans cette étude nous ne nous préoccupons pas du processus d'identification des connaissances pertinentes au métier. Ce processus peut être réalisé en utilisant des techniques issues de la représentation des connaissances. Dans le chapitre II nous fixons le cadre de cette étude en définissant ce qu'est un système d'intégration de données (SID) et les types de changements qui peuvent affecter un tel système. Nous nous intéressons ensuite dans le chapitre III aux différents langages de modélisation du schéma global d'un SID qui permettent de représenter les connaissances métier. Le chapitre IV abordera les différentes techniques de mise en correspondance de ce schéma global avec les sources de données. Nous nous intéresserons plus particulièrement à l'étude de l'impact des différents changements sur ces correspondances. Le chapitre V propose une analyse des systèmes existants suivant un certain nombre de critères que nous jugeons importants.

Chapitre II.

Système d'intégration de données

II.1 Système d'intégration

Nous allons dans cette section nous attacher à définir ce qu'est un système d'intégration et les facteurs à prendre en considération lors de la conception. Nous verrons aussi les deux principales architectures utilisées pour la mise en place d'un système d'intégration.

II.1.1 *Définition*

Comme nous l'avons précisé dans le chapitre introductif de ce manuscrit, les entreprises collectent de plus en plus d'informations sur leur environnement (clients, concurrents, marchés, partenaires...). Ces *informations* sont généralement *disséminées* au sein de plusieurs systèmes d'information. L'objectif d'un système d'intégration[2][3] est de *permettre aux utilisateurs métiers de se concentrer sur les informations pertinentes et nécessaires à la réalisation de leurs tâches sans avoir à se préoccuper de comment et où récupérer ces informations*.

Un système d'intégration est donc un système qui permet aux utilisateurs d'accéder de manière transparente à un ensemble de sources de données hétérogènes et distribuées. Par transparence, nous entendons que le système d'intégration apparaît comme une source de

données unique aux yeux des utilisateurs. Cette illusion est réalisée par l'intermédiaire d'un schéma global aux différentes sources de données qui harmonise la description des données. Les utilisateurs ne voient pas non plus que ces données sont récupérées à partir de différentes sources de données.

Classiquement un **système d'intégration** est défini comme un **triplet** $\langle G, S, M \rangle$ où G est un **schéma global**, S est l'ensemble des **sources de données** et M l'ensemble des **correspondances** qui relient le schéma global G et les sources de données S.

II.1.2 *Conception.*

La conception d'un système d'intégration nécessite la prise en compte de plusieurs facteurs. En effet, les sources de données participant au système d'intégration possèdent rarement les mêmes organisations en terme de schéma ou données. Il est donc nécessaire d'identifier le contenu des différentes sources de données afin de vérifier qu'elles sont conformes aux besoins d'intégration. Ce besoin conduit généralement à la définition du schéma global précisant le besoin informationnel.

Le contenu des sources de données identifié, il est nécessaire de l'associer au schéma global. Cette association est nécessaire afin de permettre la récupération des données à partir du schéma global. L'association des sources de données au schéma global peut être virtuelle ou matérialisée. Suivant le type d'association choisi la récupération des données pose différents problèmes. Nous verrons ces différents problèmes plus en détails dans les sections IV.3 et IV.4.

II.1.3 *Architecture.*

Un système d'intégration peut être structuré de deux façons. Soit de manière centralisée au sein d'une architecture appelée "médiateur" soit de manière décentralisée au sein d'une architecture pair-à-pair appelée Peer Data Management System (PDMS).

II.1.3.1 Médiateur

La notion de médiateur a été introduite par Wiederhold [4][5] dans les années 90. Il le définit

comme étant *une couche intermédiaire introduite entre un client et un serveur afin de rendre plus flexible leur relation*. Le client correspond généralement à des applications d'aide à la décision telle que des outils statistiques, de planification ou de reporting alors que les serveurs correspondent aux sources de données généralement représentées par des bases de données et des fichiers semi-structurés ou plat (cf. Figure 5).

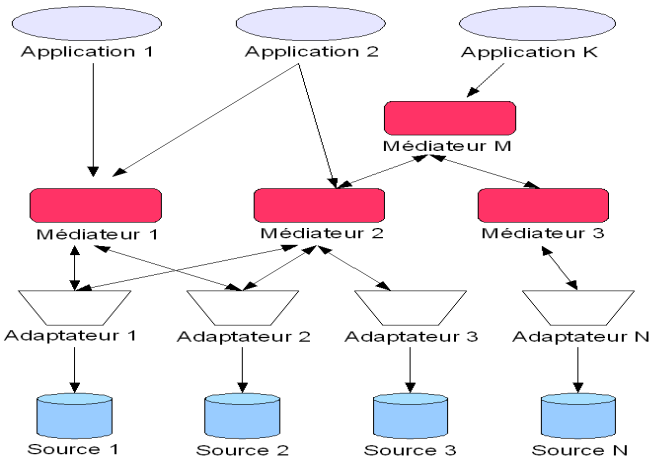

Figure 5: Architecture médiateur

Un médiateur exploite le triplet [6] $M = \langle G, S, M \rangle$ où G correspond au schéma global du médiateur qui abstrait le contenu des sources de données. L'**objectif** du médiateur dans ce contexte est donc de *reformuler une requête posée sur son schéma en utilisant les correspondances* M définies afin de récupérer les données présentes dans S. Un médiateur virtuel est un médiateur qui ne permettrait pas d'accéder, à des sources de données, mais à d'autres médiateurs.

Un médiateur est par conséquent une entité logicielle qui exploite une certaine connaissance encodée à propos d'un ensemble de sources de données afin de créer de l'information de plus haut niveau. Les connaissances encapsulées peuvent, par exemple, prendre la forme de règles d'agrégation, de transformation ou de filtrage. En aucun cas le médiateur ne contient des données. En terme d'architecture logicielle, le médiateur devrait être suffisamment simple et léger pour être administré par une personne ou un petit groupe d'experts. Un médiateur est généralement utilisé avec un adaptateur permettant d'accéder aux sources de données.

Wiederhold définit aussi des services que peut remplir un médiateur, cette liste est bien

entendu non exhaustive [7]:

- Services d'agrégation et de fusion de données,
- Services de veille sur les données et émission d'alertes lors de dépassements de seuils,
- Service de renvoi de métadonnées,
- Service de sécurité permettant de filtrer les données personnelles,

L'approche basée sur le médiateur peut être considérée comme une approche centralisée dans la mesure où il existe généralement un schéma global permettant l'intégration des sources de données. De plus, un médiateur ne prend pas en compte les aspects suivants :

- le fait qu'une seule représentation, sous la forme d'un schéma global représentant l'ensemble des métiers, n'est pas facile à construire, difficile à entretenir et peu réutilisable lors de la réalisation de tâches spécifiques aux différents métiers,
- le retrait volontaire ou involontaire d'une source de données qui rend l'utilisation du schéma global partielle, seule une partie des données représentées par le schéma est récupérable,
- les participants aux systèmes d'intégration ne veulent pas forcément partager toutes les données qu'ils possèdent et ces données peuvent être redondantes avec celles déjà représentées dans le SID,
- la difficulté de modifier les structures des sources de données intégrées sans impacter le schéma global. Toutes modifications dans la structure des sources sont susceptibles de rendre une partie des correspondances inconsistantes.

II.1.3.2 Peer Data Management System (PDMS)

L'approche PDMS [8][9] *généralise au web sémantique les systèmes d'intégration de données basés sur les médiateurs*. En effet, dans un environnement caractérisé par la distribution, les systèmes basés sur les médiateurs posent des problèmes de scalabilité de part leur organisation centralisée. Un PDMS est caractérisé de la façon suivante :

- Chaque pair possède son propre schéma visible par les pairs voisins.
- Chaque schéma, possédé par un pair, est un schéma global qui permet d'intégrer une ou plusieurs sources de données
- un pair peut avoir plusieurs rôles, serveur de données, médiateur ou client de données

Le PDMS permet à chaque pair d'avoir son propre service de données lui permettant de répondre aux requêtes par rapport à son propre schéma global. Les pairs peuvent ainsi communiquer entre eux afin d'obtenir les données souhaitées. Le PDMS permet de prendre plus facilement en compte l'ajout ou la suppression de nouvelles sources de données.
Afin de définir les relations existantes entre les sources de données et le schéma global d'un pair, mais aussi les relations entre pairs, il est nécessaire de spécifier un ensemble de correspondances. Ces correspondances peuvent être de deux natures :

- les correspondances d'intégration des sources de données ou
- les correspondances de relations sémantiques entre pairs.

Les correspondances entre pairs et sources de données sont similaires à celles utilisées au sein d'un système à base de médiateur. Elles prennent donc la forme d'un triplet $\langle G, S, M \rangle$.
Les **relations sémantiques** entre pairs peuvent être définies selon trois stratégies[9] : les *correspondances Pair à Pair, médiation de pairs ou médiation de super-pair*. Différentes méthodes peuvent être distinguées pour établir ces correspondances. Nous avons, en effet, identifié dans la littérature les méthodes proposées par Ng [10], Chang et Garcia-Molina[11], Halevy[12], Ooi[13].

- Ng associe des **mots clés** aux éléments des différents schémas.
- Chang et Garcia-Molina utilisent des **règles syntaxiques** permettant de faire correspondre les éléments d'un schéma à un autre,
- Halevy quant à lui, propose l'utilisation de **correspondances** de type GAV et LAV pour définir les relations sémantiques entres les pairs
- Ooi propose de propager les requêtes suivant les **accointances** qu'un pair a avec ses voisins.

Après avoir vu ce qu'était un SID et comment il pouvait être structuré, nous verrons dans la section suivante quels sont les changements qui peuvent survenir au sein des composants du SID. Dans un environnement qui évolue de plus en plus rapidement, le SID est amené à évoluer. Cette évolution peut être liée, soit aux changements au niveau des sources de données, soit au niveau du schéma global.

II.2 Évolution du système d'intégration de données.

Les **sources de changement** sont multiples [14]. Que ce soit au niveau des sources de données ou du schéma global, les mêmes causes s'appliquent. Une des causes les plus fréquentes est le *changement dans l'univers modélisé* du fait de son évolution naturelle [15]. De nouveaux concepts apparaissent, certains sont enrichis et d'autres disparaissent. De plus, une modélisation est rarement statique dans la mesure où l'interprétation faite de l'univers est susceptible de changer suivant les expériences ou connaissances acquises. Une autre cause de changement peut être liée à l'e*xtension ou l'amélioration du système* afin de fournir de nouvelles fonctionnalités.

II.2.1 *Au sein de sources de données*

Dans la littérature[15][16], les changements, en matière de sources de données, peuvent être soit des *changements de contenu* soit des *changements de structure*.

Les **changements de contenu** s'inscrivent dans le cadre d'une *utilisation normale des sources* de données. Ce type de changement prend généralement la forme d'ajout, de suppression ou de mise à jour des données.

Les **changements de structure**, quant à eux, sont des *modifications dans l'organisation du stockage* des données. Dans le cadre de sources de données relationnelles par exemple, les relations peuvent être supprimées, divisées, fusionnées, la cardinalité des liens avec d'autres relations modifiée. Les changements de structure peuvent donner lieu à une redéfinition de schéma ou révision de schéma.

Une **redéfinition de schéma** consiste à *remplacer le schéma existant par un nouveau schéma* avec une structure différente. La redéfinition de schéma est généralement lourde de conséquences dans la mesure où les données de l'ancien schéma devront être transformées pour être conformes au nouveau schéma. Cette transformation peut engendrer une perte de données.

La **révision de schéma** est moins brutale et permet, à partir de l'ancien schéma, d'effectuer des *changements mineurs* qui sont beaucoup moins lourds de conséquences que la redéfinition de schéma. La révision peut être considérée comme une réparation du schéma existant alors que la redéfinition est un remplacement du schéma. Les opérations peuvent être

l'ajout ou la suppression de relations, le renommage de la relation et la division ou fusion de relations. Dans le cas des attributs, les opérations sont l'ajout, la suppression, le renommage ou le changement de type.

II.2.2 *Au sein du schéma global*

Les changements au niveau du schéma global peuvent être dus à des *nouveaux besoins informationnels* ou à des *évolutions dans le domaine modélisé*. Contrairement aux changements de sources de données, il n'y a pas de changement au niveau du contenu dans la mesure où il n'y a aucune donnée de stockée au niveau du schéma global. Les seuls changements qui peuvent se produire sont des changements de structure.
Ces changements au niveau du schéma global peuvent aussi se décliner sous la forme de révision ou de redéfinition. Dans le premier cas, il s'agit de mettre à jour les éléments obsolètes, alors que, dans le deuxième cas, le schéma global est remplacé dans son intégralité.

II.3 Synthèse

Dans ce chapitre, nous nous sommes attachés à préciser les objectifs d'un SID. Nous avons aussi identifié les trois composants à savoir le schéma global, les sources de données et les correspondances entre le schéma global et les sources de données. Nous avons aussi décrit comment ces composants peuvent être organisés pour former une architecture centralisée ou décentralisée. Finalement, nous avons aussi identifié les différents types de changements qui peuvent survenir au niveau des sources de données et du schéma global. Nous allons voir dans les chapitres suivants les différentes façons d'exprimer un schéma global et d'exprimer les correspondances. L'analyse de ces différentes solutions est faite sous l'angle de la flexibilité par rapport aux changements définis plus haut.

Chapitre III.

Représentation des données et schéma global.

III.1 Introduction

Dans ce chapitre, nous nous attachons à décrire les différentes solutions pour l'expression d'un schéma global. Comme nous l'avons déjà précisé, le schéma global permet d'abstraire le contenu des sources données au sein d'un SID. Une abstraction sous forme de connaissances métiers permet une meilleure utilisation d'un tel système. Les solutions pour représenter un schéma global peuvent être regroupées en quatre catégories:

- le modèle relationnel,
- le modèle objet,
- le méta-langage XML, et
- les approches en logiques de description.

Comme nous l'avons précisé auparavant, le schéma global permet de cacher aux utilisateurs l'existence de multiple sources de données. Il représente les informations contenues dans les diverses sources de données suivant un besoin informationnel. Cette représentation doit donc permettre la prise en compte d'informations structurées de différentes façons, en plus de la

prise en compte de l'hétérogénéité des données ainsi que les différents types de changement que nous avons décrits dans le chapitre précédent.
Suivant la solution adoptée, pour chacune de ces approches, nous décrirons le processus suivi pour la définition du schéma global.

III.2 Modèle Relationnel

Le modèle Entité-Relation (ER) introduit en 1976 par Chen[17] permet une description de haut niveau des données à travers un modèle conceptuel des données généralement représenté par le diagramme Entité-Relation[18]. Nous verrons dans cette section ce qu'est un modèle relationnel et quels sont les langages qui permettent de le manipuler.

III.2.1 *Définition*

Le modèle Entité-Relation représente les concepts qui sont manipulés dans un domaine précis ainsi que les relations qui existent entre ces entités. Les entités, tout comme les relations, peuvent avoir des attributs qui permettent de les décrire. Ce modèle est ensuite décliné en un modèle logique de données, le modèle relationnel, afin d'être implémenté.
Le modèle relationnel repose sur la théorie mathématique des relations et a été proposé par Codd [19] en 70. Il permet de décrire logiquement les données et est utilisé pour implémenter physiquement, au sein d'un système de gestion de base de données quelconque, la vision retenue d'un domaine.
Un modèle relationnel est composé de relations, de domaines et d'attributs. Une relation correspond à un ensemble d'attributs qui décrit une même chose. Un attribut est composé d'un identificateur et un ensemble de valeurs définies sur un domaine. Un domaine est un ensemble de valeurs (entier, réel, booléens...) fini ou infini qu'un attribut de la relation peut prendre. Une relation, notée R est le sous-ensemble du produit cartésien $R \subseteq D_1 \times D_2 \times \ldots \times D_n$ des n ensembles $D_i, i \in \{1, \ldots, n\}$, représentant des domaines qui ne sont pas obligatoirement distincts. Une relation peut être vue comme un tableau à deux dimensions où les colonnes représentent les domaines et les lignes des tuples ou n-uplets.

Pour une relation $R(A_1 : D_1, A_2 : D_2, \ldots A_n : D_n)$, A_i est le nom d'un rôle joué par son domaine D_i.Un tuple correspondra à $r = (a_1, a_2, \ldots a_n)$ où les a_i sont des valeurs qui prennent leurs valeurs dans leur domaine D_i respectif.

Un schéma de relation désigne le nom de la relation, la liste de ses attributs ainsi que leurs domaines de définition et la/les clés de la relation. Le schéma de relation définit l'intention de la relation et les tuples l'extension de la relation.

Afin de manipuler un modèle relationnel, plusieurs langages ont été définis.

III.2.2 *Langage de manipulation*

L'algèbre relationnelle [20][21] introduite en 1970 par Codd est un langage procédural composé d'opérateurs ensemblistes (l'union, la différence et le produit cartésien) et d'opérateurs unaires (la restriction et la projection). Ces cinq opérateurs, sont utilisés pour construire d'autres opérateurs tels que l'intersection, la division, la thêta-jointure et la somme. Ces opérateurs permettent la composition d'expressions plus complexes permettant d'apporter une réponse aux requêtes posées.

Le calcul de prédicats [20][21] (sur les tuples ou sur le domaine des attributs) est, contrairement à l'algèbre relationnelle, un langage déclaratif permettant aux utilisateurs de spécifier l'ensemble de données recherchées sans avoir à dire comment les récupérer. Il existe deux sous-langages: le calcul de prédicats sur les tuples et le calcul de prédicats sur les domaines.

Le calcul de prédicats sur les tuples permet d'obtenir un sous-ensemble de tuples satisfaisant un ensemble de contraintes sur une ou plusieurs variables de tuples.

Dans [22] Codd a montré l'équivalence entre l'algèbre relationnelle et le calcul de prédicats. Ces deux langages ont grandement influencé SQL. Originellement développé par IBM dans le cadre du projet SEQUEL/XRM et System-R dans les années 70, SQL est rapidement devenu le langage utilisé *de facto* dans les bases de données relationnelles. Au départ, SQL, a été conçu pour être différent de l'algèbre relationnelle ou du calcul de prédicats. Cependant, durant son évolution, certains éléments provenant de ces deux domaines ont été intégrés afin d'étendre les fonctionnalités proposées et d'optimiser son fonctionnement. La partie correspondant à l'algèbre relationnelle permet de répondre à la question comment structurer une requête, et la partie correspondant au calcul des prédicats répond à la question comment

apporter une réponse à la requête.

Le principal inconvénient du modèle relationnel est son manque de flexibilité. En effet, *le modèle relationnel est peu adapté à la prise en compte de l'hétérogénéité des données dans la mesure où le contenu du modèle relationnel doit se conformer au modèle défini*. La prise en compte d'éléments ayant une structure différente de celle prévue par le schéma relationnel nécessite soit une transformation de l'élément afin de le rendre conforme au schéma ou la modification du schéma. Si par exemple le modèle défini qu'une l'adresse est composée de trois attributs ADRESSE, CODE_POSTAL et VILLE, il sera nécessaire de transformer les adresse stockées sous la forme d'une seule chaîne de caractères au sein d'une autre source de données. Il n'existe, par conséquent, aucune hétérogénéité au sein d'une base de données relationnelle. Nous verrons, dans la section suivante, le modèle objet qui permet plus facilement de prendre en compte l'hétérogénéité ainsi que des évolutions du schéma.

III.3 Modèle Objet

Dans cette section, nous nous attachons à décrire comment un schéma global peut être modélisé à partir d'une approche objet. Deux approches sont possibles: l'Object Definition Language Intelligent Information Integration (ODLI3)[23] et l'Object Exchange Model (OEM)[24].

III.3.1 *ODLI3 et son langage de manipulation OLCD*

ODLI3[23][25] est un langage qui étend Object Definition Language[26] et respecte les recommandations de l'ODMG (Object Data Management Group) sur la description d'objets. ODLI3 permet la description indépendante des sources de données décrites. Il introduit deux opérateurs, les relations et les contraintes d'intégrité.

Les opérateurs introduits sont l'union et l'optionnel. L'union établit une équivalence entre des structures différentes et permet de prendre en compte l'hétérogénéité des structures de données. L'optionnel, représenté par un *, permet de préciser qu'un attribut est facultatif dans

la description d'une classe.

Les relations en intention et en extension ajoutées permettent d'exprimer des relations sémantiques intraschéma et interschémas pour différents schémas sources. En effet, il est possible d'exprimer des relations de synonymie (SYN), de subsomption (BT) et d'association (RT) (deux mots utilisés ensemble dans un contexte spécifique). Pour définir par exemple que le terme T_1 de la source de données S_1 subsume le terme T_2 de la source S_2, on écrit $\langle S_1.T_1\ BT\ S_2.T_2 \rangle$.

En dernier lieu, ODLI3 rajoute des contraintes d'intégrité sous forme de règles si-alors ou de correspondances permettant d'établir les relations entre le schéma global et les sources de données.

Object Language with Complements allowing Descriptive cycles (OLCD)[27] est un langage de logique de description basé sur OLC (Object Language with Complements). Il étend Object Description Language (ODL)[28] qui est similaire aux langages terminologiques et qui permet la description de classes et de valeurs complexes à partir de valeurs atomiques. Cette extension introduit les chemins quantifiés qui permettent d'effectuer des requêtes sur une structure objet.

Il permet aussi de spécifier des contraintes sur une propriété d'un objet appartenant à une structure imbriquée. Les contraintes pouvant être appliquées sont l'existence et l'universalité ainsi que les opérateurs de comparaison. Par exemple, $Humain.Homme.nom = Jean$ spécifie que le nom de l'objet "Homme" qui est imbriqué dans le concept « Humain » doit correspondre à « Jean »

Les contraintes d'intégrité sont exprimées de manière déclarative sous forme de règles si-alors avec un quantificateur universel. L'introduction de ces contraintes permet de vérifier la validité et la cohérence du schéma et d'optimiser la récupération des résultats d'une requête.

L'opérateur union permet, quant à lui, d'exprimer les relations d'héritage, mais aussi de traduire l'opérateur optionnel introduit par ODLI3. En effet, l'opérateur optionnel, introduit en ODL, est traduit en OLCD comme étant l'union du domaine de validité de l'attribut et du domaine indéfini.

III.3.2 *OEM et son langage de manipulation OEM-QL*

OEM[24][29] peut être considéré comme une *approche orientée-objet simplifiée*. La notion d'objet s'apparente à un quadruplet composé d'un identifiant d'objet, une étiquette qui représente le nom de l'objet, de son type et d'une ou plusieurs valeurs conforme au type défini ou un ensemble d'objets $\langle Oid\ ,\ etiquette\ ,\ type\ ,\ valeur \rangle$.
Par exemple, un objet "Personne" possède une étiquette "Personne" et un ensemble de valeurs. Ces valeurs sont des sous-objets tels que le "nom" correspondant à l'étiquette, un type qui peut être une chaîne de caractères et une valeur telle que « Pierre ». OEM peut par conséquent être considéré comme étant auto-descriptif dans la mesure où un objet va encapsuler les métadonnées qui permettent de le décrire.
OEM ne peut cependant pas être considéré comme un modèle orienté-objet conventionnel, car les seules notions supportées par OEM sont l'imbrication d'objets et leur identification. Il n'existe donc pas de notion de classe, de méthode ou d'héritage. Un modèle simple a volontairement été choisi afin de ne pas contraindre, comme ferait une classe, la représentation d'éléments hétérogènes.

OEM-QL[24] est un langage de requête qui se base sur l'Object Query Language (OQL) et qui hérite de la syntaxe de SQL et de la structuration de OQL. Il permet d'interroger des modèles en OEM avec des expressions de type Select-From-Where. Cependant, il se différencie de SQL ou OQL par le traitement du OU dans une requête. En effet, lors de l'exécution d'une requête SQL ou OQL de type $R \cap (S \cup T)$, si R est vide alors aucun résultat ne sera renvoyé. En effet en SQL, l'expression serait « R.attributR = S.attributS or R.attributR=T.attributT », étant donné qu'il n'y a aucun enregistrement appartenant à R le résultat final sera un ensemble d'enregistrements vide alors que S et T sont non vides. Dans un SID, ce scénario risque de se produire très souvent et OEM-QL permet de le prendre en considération.

Ces modèle objets sont très flexibles et simples. Il est possible de faire évoluer la structure des données sans remettre en cause l'existant, ce qui n'est pas le cas du modèle relationnel. Cependant c*es modèles objets présentent l'inconvénient de ne pas pouvoir représenter la notion de schéma. En effet, ils décrivent le contenu mais ne présentent aucun résumé de comment sont structurées les données*. De ce fait lors de la récupération il est nécessaire de

parcourir tout le modèle pour savoir s'il peut répondre à une requête quelconque. Afin de résoudre ce problème de schéma, il est possible d'utiliser des "Dataguides" qui sont des schémas calculés a posteriori à partir du modèle objet mais ceci reste une opération très coûteuse. Nous verrons dans la section suivante comment XML permet de supporter des données semi-structurées tout en fournissant des informations sur le contenu mais aussi sur la structuration des données.

III.4 Méta-langage XML

XML a été développé par le W3C et permet de remédier aux inconvénients du HTML, à savoir son inadaptabilité dans l'échange de documents du fait de son manque de structuration. XML est basé sur le SGML qui est un meta-langage permettant la définition de langage d'échange de documents et de données. XML donne la possibilité aux utilisateurs de définir leurs propres balises pour la structuration des données qu'ils souhaitent échanger.

III.4.1 *Définition.*

XML est de nos jours le format standard pour la représentation de données semi-structurées et permet facilement de représenter les modèles de données OEM[20] vus précédemment.
Comme nous l'avons dit, XML permet aux utilisateurs de définir leurs propres balises afin de structurer leurs documents. Un document qui respecte les règles XML est considéré comme un document bien formé. Il est considéré comme valide s'il respecte son schéma c'est à dire la Document Type Definition (DTD) ou le XML Schéma qui a servi à définir sa structure.
Une DTD spécifie la grammaire, à savoir les éléments et attributs qui peuvent être utilisés dans un document, ainsi que les contraintes qu'ils doivent respecter. Elle permet aussi de préciser l'imbrication que doivent respecter les éléments. Les inconvénients de la DTD sont que l'utilisateur est limité dans la description des types de données. En effet, une DTD ne dispose que du type #PCDATA pour la définition d'entiers, de booléens ou de chaîne de caractères. De plus, une DTD n'est pas écrite en XML ce qui pose des problèmes au niveau du stockage de document XML. Ce qui impose d'avoir deux structures de stockage différentes,

une pour le document XML et une pour la DTD. Par ailleurs, une DTD est difficilement réutilisable du fait qu'elle n'autorise pas l'importation de tout ou partie d'une autre DTD.
XML Schéma permet, comme une DTD, de spécifier la structure d'un document XML, mais il comble les lacunes de la DTD et introduit de nouvelles possibilités. Un schéma XML est écrit en XML et permet donc l'utilisation des mêmes outils que ceux utilisés pour le traitement de fichiers XML. Il permet le typage de données et la création de types de données complexes à partir des types de base. Il introduit aussi l'héritage d'éléments, la possibilité de spécifier la cardinalité des occurrences d'éléments et les espaces de nom.
Afin d'interroger les documents décrits en XML, plusieurs langages ont été définis. Nous décrivons dans la section suivante, ces différents langages.

III.4.2 *Langages de requête pour XML*

Plusieurs propositions ont été faites en matière de langage de requêtes pour des documents semi-structurés écrits en XML. Dans [30] les différents langages de requête sont décrits, à savoir XPATH, QUILT, XMAS[31], XML-QL[20], XQL et Xquery.
XML-QL est un langage de requête déclaratif à base de règles. Il a été développé par les laboratoires de AT&T. Une requête XML-QL est composée de deux blocs CONSTRUCT et WHERE. Il est possible d'imbriquer, dans le bloc CONSTRUCT, plusieurs sous-requêtes portant sur d'autres documents XML. Le bloque WHERE permet de spécifier les éléments à récupérer, ces éléments peuvent appartenir à différents documents XML. Le bloc CONSTRUCT permet lui de formater le résultat obtenu.

XMAS pour XML Matching And Structuring language ressemble beaucoup à XML-QL. C'est aussi un langage de requête déclaratif à base de règles. Les règles prennent aussi la forme de bloc CONSTRUCT-WHERE. Cependant, XMAS gère l'agrégation et le regroupement de façon plus performante.
La clause CONSTRUCT permet de spécifier le formatage à appliquer au résultat trouvé. XMAS permet l'utilisation des opérateurs de l'algèbre relationnelle tels que la sélection, la projection et la jointure pour exprimer des requêtes plus complexes.

Xquery est le langage de requête retenu par le W3C[2]. Xquery combine les avantages des différents langages de requête pour les documents XML, notamment QUILT, XML-QL, XQL, et n'est pas basé sur des règles. Une requête Xquery correspond à une expression FOR-LET-WHERE-ORDER BY-RETURN (FLWOR)

FOR et LET permettent d'associer des variables à des éléments d'un document XML, le WHERE permet de spécifier des contraintes sur les éléments qu'on souhaite récupérer, le ORDER BY permet de classer le résultat dans un certain ordre et le RETURN permet de construire les instances d'un document XML à partir des résultats obtenus. Un exemple de requête est donné dans le tableau ci-dessous

```
Xquery


FOR $Livre in document("catalogue.xml")//Livre
WHERE $Livre/@année > 1991
RETURN <resultat>
{ $Livre/titre }
{ $Livre/auteur }
</resultat>
```

L'exemple permet de récupérer tous les titres et auteurs des livres publiés après 1991 présents dans un catalogue.

Xquery permet l'imbrication des requêtes comme avec SQL au niveau du FOR, du WHERE et du RETURN. Il introduit d'autres constructeurs tels que le COUNT, MIN, MAX, mais aussi des expressions conditionnelles comme le IF-THEN-ELSE.

Le *principal avantage de XML est sa flexibilité dans la mesure où il supporte très bien l'hétérogénéité des données* et propose plusieurs langages pour la récupération de données. Du fait de sa standardisation et des outils qui ont été développés, XML est le langage *de facto* pour la définition d'un format pivot[3]. Ce format permet de spécifier la structures des objets

2 Recommandation du 23 Janvier 2007 http://www.w3.org/TR/xquery/

3 L'utilisation d'un format pivot consiste à définir un format "universel". Celui-ci est soit totalement indépendant des applications devant être intégrées ou bien basé sur les contraintes d'une application largement "dominante". L'ensemble des données qui transitent par la solution d'intégration doit alors être conforme à ce format. L'utilisation d'un format pivot offre une couche d'abstraction supplémentaire qui contribue à un couplage plus faible des applications entre elles. Il favorise, en outre, un traitement homogène des données.

métiers, ainsi que des contrôles sur ces derniers, échangés entre systèmes d'informations ou entre partenaires. Lors des échanges, le format permet une validation par rapport à un schéma XML ou une DTD et XSLT peut être utilisé pour adapter les informations échangées. Cependant, lors de la modélisation des connaissances liées à un métier, il est nécessaire de préciser la sémantique des choses et *XML ne permet que la description syntaxique*. Il est par exemple difficile d'associer une signification à l'imbrication des éléments. Elle peut représenter des caractéristiques ou des relations. Nous verrons dans la section suivante les logiques de description qui permettent de spécifier la sémantique des choses.

III.5 Logique de description

Cette section est consacrée aux différentes logiques de description qui ont été utilisées pour modéliser un schéma global pour des systèmes d'intégration. Nous donnerons un bref historique avant de décrire succinctement CLASSIC, LOOM, CARIN-ALN.

III.5.1 *Historique*

Les réseaux sémantiques introduits par Quillian[32] en 1966 sont un formalisme intuitif de représentation et d'organisation de connaissances. Il consiste à *identifier des noeuds et à définir des liens représentant les relations entre les noeuds définis. À chaque noeud et lien définis est associée une étiquette.* Le problème des réseaux sémantiques est paradoxalement le manque de sémantique. Il est en effet difficile de différentier les liens structurels des liens d'assertions, et donc difficile de différencier les valeurs possibles des nœuds du réseau et les valeurs réelles.

Pendant que la communauté des réseaux sémantiques essayait de rendre les réseaux sémantiques plus sémantiques, Minsky[33] proposa en 1975 une nouvelle façon d'organiser et de structurer les connaissances : les frames. *Un frame est un ensemble d'attributs ayant une, plusieurs ou aucune valeur suivant les éléments représentés. En plus des attributs et des valeurs, les frames peuvent posséder une ou plusieurs facettes qui sont des contraintes en terme de valeurs ou cardinalité.*

Brachman[34] en 1977 analyse les réseaux sémantiques en 4 niveaux : le niveau conceptuel, le niveau logique, le niveau linguistique et le niveau de l'implémentation. Suite à cette analyse, il explique qu'il manque un niveau crucial : le niveau épistémologique. Suite à son analyse, Brachman propose un réseau d'héritage structurel qui est un formalisme basé sur les notions de concepts, de rôles et de description structurelle pour représenter les connaissances. Les concepts peuvent être organisés en une taxonomie où les fils héritent de toutes les caractéristiques des parents. De plus, Brachman définit deux types de concepts : les concepts génériques pouvant être utilisés pour définir des ensembles et les concepts individuels qui sont des ensembles à un élément. Les concepts génériques peuvent être divisés en concepts primitifs et concepts définis.

La proposition de Brachman a été implémentée en 1978, ce qui donna naissance à KL-ONE, l'ancêtre de la logique de description. KL-ONE est le premier système à introduire une représentation de connaissances terminologiques et la représentation des connaissances assertionnelles qui n'a été détaillée que superficiellement.

Plusieurs systèmes se sont inspirés de KL-ONE, on peut citer quelques-uns KANDOR, MESON, LOOM, SPHINX, CLASSIC.

D'autres travaux ont tenté de développer la notion de représentation assertionelle KL-TWO, NIKL.

III.5.2 *CLASSIC*

CLASSIC[35] est un descendant de KL-ONE. Il a été conçu afin de *satisfaire les besoins pratiques de ses utilisateurs et afin de garantir un comportement prévisible du système*. C'est un langage de type ALNFIh[4].

Les principales caractéristiques de CLASSIC sont les suivantes :

- le calcul de subsomption,
- la classification des individus,
- les vérifications de consistance,
- l'utilisation de règles.

CLASSIC possède une expressivité restreinte dans la mesure où les concepteurs du système

4 ALNFIH est un ensemble de lettres AL, N, F, I, H qui permettent de préciser l'expressivité d'un langage en logique des descriptions. Le lecteur intéressé peut se référer à [36] ou à http://en.wikipedia.org/wiki/Description_logic qui contient un tableau récapitulatif des degrés d'expressivité de la logique des descriptions.

ont volontairement écarté certains constructeurs. En effet, des constructeurs comme OR et NOT n'ont pas été implémentés. L'arbitrage entre expressivité et performance a eu pour effet de rendre CLASSIC moins expressif, mais plus performant en terme de tests de subsomption, de classification et d'instanciation, ce qui explique son utilisation dans de nombreux projets.

III.5.3 *LOOM*

LOOM[37] est un *langage terminologique très expressif*. Ce langage a été développé par l'université de Californie dans les années 80 pour faciliter le développement et la maintenance d'applications à base de modèles. Contrairement à CLASSIC, les concepteurs de LOOM ont choisi de doter ce dernier d'un grand nombre de constructeurs afin de permettre une grande expressivité. Il est de type ALCQRIFO. La justification d'un tel choix, selon les concepteurs, est liée au fait que la réalité est tellement complexe que pour la modéliser il est nécessaire d'avoir le plus grand nombre de constructeurs possible[38][39]. La grande expressivité de LOOM présente un certain avantage au niveau de la modélisation, mais débouche sur une incomplétude des procédures d'inférence.

III.5.4 *CARIN-ALN*

CARIN[40] est un langage à base de *règles de Horn étendu avec de la logique de description*. La logique de description utilisée est ALN. CARIN-ALN permet la description de concepts atomiques et des concepts plus complexes sous la forme Concept := définition. Les concepts complexes sont uniques. CARIN-ALN autorise aussi la conjonction, la restriction de valeurs, la cardinalité et la négation. Les règles de CARIN-ALN sont de la forme $\forall \bar{X}[p_1(\bar{X}_1)\ldots p_n(\bar{X}_n)\Rightarrow q(\bar{Y})]$ où $\bar{Y}\subseteq\bar{X}_1\cup\ldots\cup\bar{X}_n$. Les concepts pouvant être des atomes d'une règle sont : $A(X), \neg A(X), (\geq n\,R), (\leq n\,R) ou \forall R_1 \forall R_2 \ldots \forall R_k . D(X)$. Les règles peuvent être utilisées pour définir des relations n-aires.

III.5.5 *OKBC*

OKBC[41] pour Open Knowledge Base Connectivity est une API pour les systèmes de représentation de connaissances. Cet API permet de créer une interface générique pour l'accès

à plusieurs systèmes de représentation de connaissances assertionnelles. Cette interface générique est composée d'un modèle de connaissances OKBC composé de constantes, de classes, de frames, de propriétés, de facettes et d'une base de connaissances. Associées à ce modèle, un certain nombre de fonctions ont été développées afin de permettre la manipulation du modèle de connaissance OKBC pour interroger les systèmes interfacés.

III.5.6 *Synthèse*

Les années 80-90 ont connu d'autres systèmes comme K-REP[42], BACK[43] basés sur la logique de description. Le point commun entre ces différents systèmes est qu'ils utilisent l'algorithme de subsomption structurel[44]. Cet algorithme permet de tester la subsomption entre différents concepts à partir des expressions qui les composent. Cet algorithme est performant, il possède une complexité polynomiale pour des logiques de description peu complexes. Pour des systèmes plus complexes, les tests de subsomption donnent des résultats incomplets[45], ce qui explique les choix des concepteurs de CLASSIC. Au début des années 90, l'introduction d'algorithmes à base de tableaux a rendu le calcul de satisfiabilité possible pour des systèmes de logique de description sophistiqués. En effet, l'algorithme de tableau permet aux nouveaux classifieurs tels que DLP[46], FACT[47] ou RACE[48] d'améliorer le compromis entre expressivité et performance.

III.6 Web sémantique

Avec l'essor du web sémantique, le besoin d'avoir une sémantique bien définie sur laquelle il est possible de raisonner est devenu un enjeu majeur. En effet, avant l'utilisation des logiques de description dans le cadre du web sémantique, l'utilisation de RDF[49][50], RDFS[51], pour la représentation des connaissances était très répandue. L'inconvénient principal des modélisations en RDF est qu'elles sont peu expressives. *Il est effectivement impossible d'exprimer en RDF ou en RDFS la conjonction, la disjonction ou la négation. En plus du manque d'expressivité de RDF, RDFS, il est difficile de raisonner sur une telle représentation.* Du fait de ce manque d'expressivité et de possibilité de raisonnement, d'autres

langages de modélisation ont été définis.

III.6.1 *DAML+OIL*

DAML+OIL est issu de deux projets, DAML[52] un projet initié par la DARPA et OIL[53] initié par un groupe de chercheurs européens. DAML s'appuie sur les standards W3C tels que XML, RDF, SHOE[36] pour définir un cadre unifié. OIL définit un ensemble d'expressions formelles qui peut être traduit dans une logique de description de type SHIQ[36]. Le S de SHIQ correspond à ALC auquel on rajoute les rôles transitifs. H correspond aux hiérarchies de rôles, I les rôles inverses et Q les restrictions numériques.

III.6.2 *OWL*

OWL[54] est une révision du langage DAML+OIL. Il définit 3 niveaux d' expressivité OWL Lite, OWL DL et OWL Full. OWL Lite permet la construction d'ontologies simples, c'est-à-dire une ontologie composée de contraintes simples et de taxonomies. OWL DL est quant à lui plus expressif grâce à l'utilisation de constructeurs et de mécanismes d'inférences issues de la logique de description. OWL Full permet, quant à lui, la plus grande expressivité sans aucune garantie de performance. OWL est fondé sur DAML+OIL. Nous nous intéresserons plus particulièrement à OWL-DL dans la mesure où il propose le meilleur compromis entre expressivité et performance. OWL-DL est traduisible en SHOIN(D)[55] qui correspond à une logique de description avec des constructeurs de base ALC auquel on rajoute les rôles transitifs (+R), H correspond aux hiérarchies de rôles, I les rôles inverses, N les restrictions numériques non qualifiées, et (D) correspondant aux domaines concrets. OWL-DL est un bon compromis entre expressivité et décidabilité.

III.7 Synthèse

Dans ce chapitre nous nous sommes attachés à décrire les différentes solutions pour la construction d'un schéma global. Le modèle relationnel présente l'avantage d'être facilement

manipulé par des outils d'interrogation comme SQL. En raison de son utilisation dans le monde industriel, beaucoup d'utilisateurs sont déjà sensibilisés à la conception et l'exploitation du modèle relationnel. Cependant, le *modèle relationnel présente l'inconvénient d'être peu flexible en matière de prise en compte de l'hétérogénéité de structures de données, mais aussi en matière de prise en compte du changement structurel*.

Le modèle objet permet une meilleure prise en compte de l'hétérogénéité des structures de données. Il est en effet possible d'intégrer au sein d'un schéma global des sources de données structurées, mais aussi des données semi-structurées. Cependant, du fait de la semi-structuration, *il est difficile d'appliquer les techniques d'optimisation issues du domaine des bases de données pour accélérer le processus de récupération de données. En effet, il est très difficile de savoir avant le traitement du modèle s'il permet de répondre à une requête.*

XML permet non seulement de structurer les données mais permet aussi de fournir des informations sur leurs structurations. Il présente l'avantage d'être flexible et de proposer un standard en termes de modélisation et d'interrogation. Il est possible de vérifier si un document est bien formé et valide par rapport à sa DTD ou son schéma Cependant, *il présente l'inconvénient de n'être qu'une description syntaxique des données.* Afin d'associer une sémantique à XML il est nécessaire d'avoir recours à d'autres méta-langages comme RDF/RDFS ou OWL.

La logique de description permet de vérifier la consistance d'un modèle mais présente l'inconvénient, pour des langages riches, d'être coûteux en termes de calcul de consistance. Les algorithmes de type tableaux ont permis d'accélérer ce processus de calcul et l'essor du web sémantique conduit au développement de nouveaux langages transposables en logique de description.

Cependant la définition d'un schéma global pour abstraire les sources de données ne suffit pas pour créer un SID. Il est aussi nécessaire de relier le schéma global aux sources de données. Cette mise en correspondance, selon qu'elle va des sources de données vers le schéma global ou inversement, présente divers avantages et inconvénients. Nous verrons donc dans le chapitre suivant les différentes techniques proposées pour relier un schéma global aux sources de données.

Chapitre IV.

Gestion des correspondances

IV.1 Introduction

Un schéma global permet d'exprimer le contenu des sources de données afin que l'utilisateur puisse en prendre connaissance. Seul, il ne permet pas la récupération de données. Il est donc nécessaire de relier le schéma global aux sources de données. La relation entre sources de données et schéma global se fait par l'intermédiaire de correspondances. Celles-ci peuvent être **matérialisées** ou **virtuelles**[56]. Suivant l'approche suivie pour leur expression, elles peuvent être plus ou moins sensibles aux changements qui se produisent au niveau du schéma global ou au niveau des sources de données. Il est nécessaire pour assurer le bon fonctionnement du SID de veiller à la validité des correspondances. Nous verrons, dans ce chapitre, les différentes approches existantes pour l'expression des correspondances, à savoir l'approche matérialisée et l'approche virtuelle. Nous verrons aussi l'impact qu'ont les changements sur ces différentes correspondances.

IV.2 Notion de vue

Une vue[20][57] est un ensemble de données qui n'est pas explicitement stocké dans la base de données, mais recalculé à partir de sa définition à chaque fois que la vue est utilisée. *Une vue matérialisée consiste à enregistrer le résultat d'une vue en base de données sous forme d'une table.*

Les vues permettent de :

- masquer aux utilisateurs les données confidentielles en ne leur donnant accès qu'à une partie des données (celles contenues dans les vues),
- simplifier l'utilisation de requêtes complexes en enregistrant la définition de la requête,
- fournir une vision différente des mêmes données suivant les utilisateurs,
- garantir l'indépendance logique des données.

Une vue peut être utilisée comme un raccourci à des requêtes complexes. Il est en effet possible de stocker la définition de la requête complexe comme une vue pour des utilisations ultérieures.

Les vues permettent de garantir l'indépendance logique des données présentes dans la base de données. En effet, il est possible de définir une vue, accessible par les applications ou les utilisateurs, masquant, tout ou partie, du modèle conceptuel et éventuellement certaines évolutions du schéma. Imaginons qu'une relation R, utilisée par une ou plusieurs applications, soit restructurée en R' et R''. Il est possible, pour préserver les applications d'un tel changement, de définir une vue pour ces applications comme étant l'union de R' et R'' afin de retrouver l'ancienne structure.

IV.2.1 *Impacts des changements sur les vues*

L'utilisation de vues au sein de sources de données est une pratique très courante. Elle permet de personnaliser l'accès à l'information, mais aussi l'optimisation des performances. Tout changement, que ce soit au niveau des données ou de la structure, n'est pas sans influence sur les vues.

IV.2.1.1 Changement de contenu

En effet, des changements de contenu génèrent, dans le cas de vues matérialisées, un **besoin de réactualisation** des données matérialisées. Cette réactualisation peut se faire par une mise à jour des données de manière **incrémentale** ou par une **rematérialisation** de l'ensemble des données de la vue à partir de sa définition. Il est évident que la rematérialisation est facile à réaliser, mais est coûteuse en termes de traitement. La mise à jour incrémentale est, quant à elle, plus complexe à réaliser et peut être plus performante. Dans le cas de vues non matérialisées, l'ajout ou la modification de contenu n'a aucune influence, car le contenu de la vue sera recalculé à chaque appel de cette dernière.

IV.2.1.2 Changement de structure

Dans le cas de changement de structures, les deux types de vues sont impactés. Il sera nécessaire de **mettre à jour les définitions** des vues afin de refléter les changements de la source de données à laquelle elles appartiennent. Différents effets peuvent se produire suivant le type de changement(s) effectué(s). Afin de mieux comprendre l'impact des changements sur les vues, nous les verrons dans la section IV.5 après avoir décrit les différents types d'approches d'intégration.

IV.3 Approche matérialisée

IV.3.1 *Entrepôt de données*

Les entrepôts de données sont de nos jours considérés comme un composant essentiel des systèmes d'information décisionnels. Ils apportent une aide à la prise de décision en donnant une vue globale des données qui sont présentes dans les bases de données transactionnelles. W.H. Inmon, un des fondateurs du concept d'entrepôt de données, le définit comme étant **orienté sujet, historisé, non volatile et intégré** pour la prise de décisions [58][59]. *Un entrepôt de données est un ensemble de vues matérialisées à partir de différentes sources de*

données. En quelque sorte un entrepôt donne une vue orientée métier des données présentes dans des sources de données potentiellement distribuées et hétérogènes.

Avant d'arriver à un tel résultat, il est nécessaire d'identifier les besoins d'analyse des utilisateurs ainsi que les sources de données pertinentes. À partir de ces besoins, un schéma global peut être défini. Ce schéma peut être décrit en relationnel ou XML. L'entrepôt de données, répondant à un schéma donné, est ensuite *peuplé suivant un processus appelé ETL* (Extraction, Transformation and Loading). Le processus ETL consiste à extraire les données des différentes sources, à transformer ces dernières pour qu'elles soient cohérentes entre elles et finalement à les charger dans l'entrepôt de données [60][61].

L'approche matérialisée présente l'avantage d'être performante dans la mesure où ce qui est interrogé est une base de données. Il est donc possible d'utiliser les nombreux outils d'optimisation de performance déjà développés. Elle présente cependant l'inconvénient de nécessiter un processus d'ETL qui se fait généralement en *batch* et a un impact sur la f*raicheur des données disponibles*. De plus, on est vite confronté à de très *gros volumes de données* du fait de leur historisation. En plus de ces aspects de fraîcheur et de volume de données, toute évolution dans les sources de données a un impact sur le schéma de l'entrepôt de données. En effet, dans un contexte où les différentes sources de données appartiennent à des entités (service, département ou entreprise) autonomes, une architecture d'intégration matérialisée introduit une très grande rigidité par rapport aux évolutions des sources de données. Toute évolution de la structure d'une des sources de données peut potentiellement rendre le schéma de l'entrepôt inconsistant par rapport à la source de données, mais aussi par rapport aux procédures ETL définies pour cette source de données. L'*évolution du schéma global, pour la prise en compte de nouveaux besoins utilisateurs, est aussi très difficile*, car il peut être nécessaire de re-modéliser l'entrepôt de données et le re-matérialiser. Les entrepôts de données, centralisant des grands volumes de données (plusieurs Gigaoctet voire Téraoctet), nécessitent la mise en place de *datamarts*[5] afin de permettre un accès optimisé à une partie des données répondant à un besoin spécifique. Il est pas rare que les *datamarts* pour des données financière, relatif au personnel ou à la production soient mises en place. La mise en place de ces datamarts soulèvent des problématiques d'optimisation afin de permettre à un grand nombre d'applications d'accéder à l'entrepôt de données.

5 Le terme *Datamart* (littéralement magasin de données) désigne un sous-ensemble du *datawarehouse* contenant les données du *datawarehouse* pour un secteur particulier de l'entreprise (département, direction, service, gamme de produit, etc.).

Depuis peu, les solutions dites MDM (Master Data Management) on fait leur apparition. *Le MDM regroupe l'ensemble des données dites "de base" (ou Master Data) au sein d'un référentiel. Celui-ci servira alors de modèle lors de la mise à jour des systèmes ou bases de données au sein d'une organisation.* Ces solutions sont différentes des entrepôts de données dans la mesure où elles ne se contentent pas de centraliser les données, elles permettent de propager les mises à jour des données qu'elles contiennent sur les autres systèmes. En effet, la mise à jour des enregistrements, faisant référence à un objet métier, au sein d'une base de données, engendre un ensemble de mises à jour au sein d'autres bases de données contenant des informations relatives au même objet métier.
Dans la section suivante, nous verrons une alternative qui consiste à laisser les données dans les sources de données et à interroger uniquement les sources qui sont pertinentes pour répondre aux requêtes.

IV.4 Approches virtuelles

L'approche virtuelle **ne matérialise pas** les données d'un schéma global mais construit ce dernier et le relie aux données. Le lien entre schéma global et sources de données prend la forme de r*ègles de correspondances définies en terme de vues*. Deux approches **classiques** et deux approches hybrides expriment ces règles : Le *global as view* [62][63][64][65] exprime les éléments du schéma global en fonction des sources de données. Le *local as view* [62][63][64][65] suit l'approche inverse en exprimant les sources de données en fonction des éléments présents dans le schéma global. Les **deux approches hybrides**, *global local as view*[66] et *both as view*, permettent d'exprimer les correspondances dans les deux sens du schéma global vers les sources de données et des sources de données vers le schéma global. Nous présentons dans les sections suivantes ces différentes approches.

IV.4.1 *Global as view (GAV)*

L'approche GAV consiste à définir *les éléments du schéma global comme un ensemble de vues sur les sources de données*. Les correspondances sont exprimées sous la forme

d'assertions prenant la forme $\langle r, V \rangle$ [6] où r correspond à un concept de G et V à une vue sur les sources de données. Un entrepôt de données suit une telle approche à la différence que les données ne seront pas matérialisées dans l'approche virtuelle, que la modélisation est orientée vers l'analyse de données et pas vers la récupération de données. A partir du schéma global, les utilisateurs peuvent formuler des requêtes. Elles seront converties en sous-requêtes sur les différentes sources de données concernées. La reformulation de requêtes sur le schéma global en sous requêtes se fait en remplaçant les éléments de la requête globale par leur définition.

IV.4.1.1 Expression des correspondances

Les correspondances GAV consistent à exprimer les sources de données en fonction des concepts présents dans le schéma global. Les correspondances sont illustrées dans la Figure 8. Dans cet exemple, nous considérons que le schéma global ainsi que les sources de données sont exprimés en terme de relations. Soient trois sources de données contenant les relations suivantes :

S_1
$Professeur(idp, nom, prenom, date\_naissance)$

S_2
$Article(ida, titre, conference, date, auteur)$

S_3
$Conference(idc, nom, specialité, impact\_factor)$

Figure 6: Sources de données.

Pour un schéma global qui décrit comment est valorisée la recherche des professeurs, les relations sont les suivantes :

$Auteur(nom, prenom)$
$Publication(titre, date, auteur)$
$ConferenceDePublication(nom\_article, nom\_conference, specialité, impact\_factor)$

Figure 7: Schéma global.

Afin de relier le schéma global aux sources de données, les correspondances suivantes doivent être définies.

6 la notation $\langle r, V \rangle$ pour le GAV veut simplement dire que r correspond aux concepts et V aux relations.

$$Auteur(nom, prenom) \leftarrow S_1.Professeur(nom, prenom, date\_naissance)$$
$$Publication(titre, date, auteur) \leftarrow S_2.Article(ida, titre, conference, date, auteur)$$
$$ConferenceDePublication(nom\_article, nom\_conference, specialité, impact\_factor) \leftarrow S_3.Conference(idc, nom, specialité, impact\_factor), S_2.Article(ida, titre, conference, date, auteur)$$

Figure 8: Correspondances GAV.

La relation auteur dans le schéma global est définie par la projection (Π). Cette instruction permet de sélectionner un ensemble de colonnes dans une table. Dans notre exemple, « Professeur » appartient à S_1 et nous souhaitons récupérer le nom et le prénom : la projection s'écrit de la manière suivante;

$$\Pi_{nom, prenom}(S_1.Professeur)$$

Dans le cas de la relation ConferenceDePublication, il s'agit d'une jointure sur les sources S_2 et S_3 suivie d'une projection sur les attributs qui la composent.

$$\Pi_{nom, specialité, impact\_factor, titre}(S_{2\text{titre}} |><|_{nom} S_3)$$

Une fois les correspondances définies, celles-ci sont utilisées pour relier le schéma global aux sources de données, mais aussi pour reformuler les requêtes. Nous allons voir dans la section suivante comment sont ré-écrites les requêtes pour qu'elles soient compréhensibles par les sources de données.

IV.4.1.2 Réécriture de requêtes

Le processus de ré-écriture de requêtes sert à transformer une requête, posée sur le schéma global, en une ou plusieurs requêtes, sur les sources de données, à partir des correspondances définies[67][68]. Dans le cas du GAV, ce *processus consiste à remplacer les termes par leurs définitions*. Par exemple, si on souhaite trouver les titres et date de publication des articles publiés par 'Dupond' dans les actes de la conférence 'ECAI' et dans la spécialité 'Système Multi Agents; nous aurons l'expression suivante :

$$requête(titre, date, spécialité) \leftarrow S_3.Conference(idc, 'ECAI', 'Système\ Multi\ Agent', impact\_factor), S_2.Article(ida, titre, 'ECAI', date, 'Dupond')$$

Figure 9: Requête GAV reformulée.

Il est alors possible de remplacer les relations du schéma global pour obtenir l'expression de la Figure 10. Cette expression peut être utilisée pour le calcul des enregistrements qui satisfont

les contraintes définies sur les ensembles de données correspondant aux différentes vues.

$$requête(titre, date, spécialité) \leftarrow Publication(titre, date, 'Dupond'),$$
$$ConferenceDePublication('ECAI', 'Système Multi Agent', impact\_factor, nom\_article)$$

Figure 10: Requête GAV.

L'approche GAV est très bien *adaptée pour des environnements où les sources de données sont stables*[63][65]. C'est-à-dire qu'il y a très peu voire pas d'évolution dans leurs structures. Dans le cas contraire, l'ajout/suppression ainsi que leurs modifications entraînent la redéfinition des correspondances entre le schéma global et les sources de données. Cette tâche de mise à jour des correspondances peut être ardue dans la mesure où il sera nécessaire de détecter et d'identifier les changements effectués au niveau des sources de données. En plus de les identifier et de les détecter, il sera nécessaire d'évaluer leurs impacts sur les correspondances. Afin de prendre en compte les environnements dynamiques, une autre solution a été proposée. Nous la verrons dans le chapitre suivant. Une deuxième approche proposée pour l'intégration de données est le Local-as-view, qui permet d'exprimer le schéma global en fonction des sources de données.

IV.4.2 *Local as view (LAV)*

L'approche LAV consiste à décrire *les sources de données en fonction des éléments du schéma global*. Elle fait le cheminement inverse que celui adopté par le GAV. Nous verrons dans cette section comment sont exprimées les correspondances en LAV et les différents algorithmes qui ont été proposés pour la reformulation de requêtes.

IV.4.2.1 Expression des correspondances

Les correspondances sont toujours décrites sous forme d'assertions ayant la forme $\langle r, V \rangle$ [7] où r représente cette fois-ci les relations des sources de données et V les vues sur le schéma global. Pour les besoins d'illustration, nous reprenons l'exemple de la valorisation de la recherche des professeurs et nous rajoutons les informations suivantes :

- S_1 ne contient que des auteurs habitant en France,

7 La notation $\langle r, V \rangle$ pour le GAV et le LAV peut être confuse, cependant elle est du domaine et veut simplement dire que r correspond aux concepts ou relations et V aux relations ou concepts si on suit une approche GAV ou LAV respectivement

- S_2 ne contient que les publications de 2006,
- la relation auteur du schéma global possède un attribut supplémentaire, le pays de résidence. Cet attribut pays de résidence est aussi présent dans S_2 .

Pour le schéma global défini plus haut dans la Figure 7, nous avons les correspondances suivantes :

$$S_1(nom, prenom, pays) \leftarrow Auteur(nom, prenom, pays), Publication(titre, date, auteur),$$
$$nom = auteur, pays = 'France'$$

$$S_2(titre, nom\_conference, date, auteur, pays) \leftarrow Auteur(nom, prenom, pays),$$
$$Publication(titre, date, auteur)$$
$$ConferenceDePublication(nom\_article, nom\_conference, spécialité, impact\_factor)$$
$$nom = auteur, titre = nom\_article, date > 2005, date < 2007$$

$$S_3(nom\_conference, spécialité, impact\_factor) \leftarrow$$
$$ConferenceDePublication(nom\_article, nom\_conference, spécialité, impact\_factor)$$

Figure 11: Correspondances LAV.

À partir de ces correspondances, nous pouvons dire que S_1 contient les données relatives à la relation "Auteur" qui réside en France. S_1 contient aussi un sous-ensemble de données de la relation "Publication" à savoir le nom des auteurs.

IV.4.2.2 Réécriture de requêtes

Tout comme l'approche GAV, l'utilisateur peut poser des requêtes sur le schéma global, mais la reformulation de requêtes est moins facile. La reformulation de requêtes pour une approche LAV s'apparente au problème de *ré-écriture de requêtes à partir de vues*. *La réécriture de requêtes dans une approche LAV est plus difficile qu'un simple remplacement de définition comme dans l'approche GAV*. Le principe consiste à trouver un ensemble de requêtes exprimées en fonction des sources de données et des vues exprimées en fonction du schéma global. Dans le cas du LAV, la réécriture de requête peut être soit équivalente, soit fournir le maximum d'informations sans toutefois être équivalente. Nous décrivons rapidement, dans cette section, le principe suivi par trois algorithmes qui ont été proposés.

IV.4.2.2.1 *« Bucket » algorithm*

Cet algorithme ne peut être utilisé qu'avec des requêtes et des sources exprimées en requêtes conjonctives. *Le principe consiste à découper une requête posée sur le schéma global en sous-objectifs*[63][67][68]. *Chaque sous-objectif est un concept du schéma global et ne correspond pas aux restrictions qui sont appliquées sur ces derniers*. Pour chacun des concepts, un bucket (panier) est créé. Chaque panier va contenir les vues qui répondent partiellement ou de manière complète au sous-objectif. Dans le cas de réponses partielles, la condition d'inclusion dans le panier, est qu'il doit exister une ou plusieurs vues permettant de compléter la réponse. De plus, ces vues doivent pouvoir être unifiées grâce aux variables qu'elles contiennent. Une fois toutes les vues potentielles identifiées, l'algorithme procède au produit cartésien des paniers. Le résultat de ce produit cartésien donne toutes les combinaisons de vues pouvant potentiellement apporter une réponse à la requête globale. L'ensemble de ces combinaisons est ensuite évalué pour déterminer leur compatibilité. Par exemple, si on veut, à partir du schéma global défini, trouver les auteurs américains ayant publié un article dans le domaine informatique nous aurons la requête suivante :

$$q(nom) :- Auteur(nom, prenom, pays),$$
$$ConferenceDePublication(nom\_article, nom\_conference, spécialité, impact\_factor),$$
$$pays = US, spécialité = informatique$$

Figure 12: Requête LAV.

Les sous-objectifs sont Auteur et ConferenceDePublication et les contraintes (pays=US et spécialité = informatique) sont utilisées lors de l'évaluation de la pertinence des vues. Par exemple, la vue S_1 n'est pas pertinente, car elle ne contient que les données relatives aux auteurs français. La seule vue autorisée dans le panier correspondant au sous-objectif Auteur est la vue S_2. Dans le cas du sous-objectif ConferenceDePublication, deux vues sont possibles S_2 et S_3. Il existe dans notre cas deux reformulations possibles, soit S_2 seule soit S_2 et S_3. S_2 seule ne suffit pas dans la mesure où elle renvoie un ensemble de données plus important que celui demandé. En effet, il n'est pas possible d'appliquer une contrainte sur la spécialité. La reformulation de q en requêtes sur les sources de données q' est la suivante :

$$q'(nom) :- S_2(titre, nom\_conference, date, auteur, pays),$$
$$S_3(nom\_conference, spécialité, impact\_factor)$$
$$spécialité = informatique, pays = US$$

Figure 13: Requête LAV reformulée.

L'inconvénient de cet algorithme est qu'en présence de nombreuses vues, le temps de calcul est relativement important du fait de l'évaluation de toutes les combinaisons de vues possibles. L'algorithme **MiniCon** est une amélioration de l'algorithme Bucket dans le sens où, au lieu de considérer le produit cartésien des sous-objectifs possibles, *il ne prend que les sous-objectifs qui se chevauchent*[63]. Les deux étapes dans la réécriture de requête sont le calcul des sous-objectifs et la réécriture de la requête. Dans la première étape de calcul des sous-objectifs, l'algorithme va aussi s'intéresser aux variables de jointure de chaque sous-objectif. Seuls les sous-objectifs ayant une ou plusieurs variables en commun sont conservés.

IV.4.2.2.2 *« Inverse rule » algorithm*

Le principe de cet algorithme est de *définir des règles qui permettent d'inverser la définition d'une vue*[67][63]. Ces règles permettent alors de déterminer les tuples de la source de données à partir de ceux des vues.

Prenons un exemple à partir de la description de S_1 simplifiée.

$$S_1(nom, titre) \leftarrow Auteur(nom, ida), Publication(titre, ida)$$

Les règles d'inversions f_1 permettent de dire que les tuples des relations Auteur et Publication sont de la forme (nom, X) et $(titre, Y)$ respectivement, et que, pour obtenir les tuples de la vue $X = Y$.

$$Auteur(nom, f_1(nom, X)) :- S_1(nom, X)$$
$$Publication(titre, f_1(titre, Y)) :- S_1(titre, Y)$$

Figure 14: Règle d'inversion.

Supposons que les tuples de la vue S_1 sont :

$$Publication : \{('Intégration\ de\ données', f_1(Jean, 'Intégration\ de\ données')),$$
$$('Web\ Sémantique', f_1(Pierre, 'Web\ Sémantique')),$$
$$('Intégration\ de\ données', f_1(Bob, 'Intégration\ de\ données'))\}$$

$$Auteur : \{(Jean, f_1(Jean, 'Intégration\ de\ données')),$$
$$(Pierre, f_1(Pierre, 'Web\ Sémantique')),$$
$$(Bob, f_1(Bob, 'Intégration\ de\ données'))\}$$

Figure 15: Résultat de l'algorithme d'inversion.

et que la requête posée sur le schéma est :

$$q(nom) :- Auteur(nom, ida), Publication('Intégration\ de\ données', ida)$$

À partir des règles inverses, on obtient :

En exécutant la requête sur ces extensions, nous pouvons dire que la réponse à la requête est 'Jean' et 'Bob'

$$\{(Jean, 'Intégration\ de\ données'),\ (Pierre, 'Web\ Sémantique'),\ (Bob, 'Intégration\ de\ données')\}$$

L'approche LAV permet d'apporter une réponse aux inconvénients de l'approche GAV. En effet, l'ajout et la suppression de sources de données n'influent pas sur le schéma global. L'ajout de nouvelles sources de données consiste à définir en fonction du schéma global la source de données. On fournit ainsi un ensemble de vues supplémentaires permettant de répondre aux requêtes sur le schéma global sans le rendre inconsistant. La reformulation de requête est plus compliquée qu'avec une approche GAV mais il existe des algorithmes qui permettent de le faire.

IV.4.3 *Global local as view (GLAV)*

Le Global local as view[63][66] est une *extension du LAV* permettant la transformation des correspondances, définies suivant une approche LAV, en GAV. Plus précisément, il est possible de convertir un SID suivant une approche LAV en un SID suivant une approche GAV[69] si :

- le schéma global est exprimé avec un formalisme relationnel autorisant les dépendances de type inclusion,
- les sources de données sont aussi exprimées avec un formalisme relationnel, dans le cas contraire l'utilisation d'adaptateurs sera nécessaire,
- les vues sont exprimées en requêtes conjonctives,
- la requête exprimée sur le schéma global est une requête conjonctive.

Le principe consiste à définir de nouvelles relations dans le schéma global. Ces relations servent à définir des règles de transformation permettant de convertir les correspondances écrites en LAV en correspondances GAV.

Les correspondances sont exprimées de la même façon que dans l'approche LAV, à savoir $\langle r, V \rangle$ où r correspond à une source de données et V à une vue sur le schéma global. Le

GLAV étant une généralisation du LAV, il souffre des mêmes inconvénients que ce dernier. Les correspondances sont toujours exprimées en LAV et un ensemble de règles à été ajouté pour permettre de transformer en GAV les correspondances exprimées en LAV. De ce fait toute modification du schéma global est lourde de conséquence.

IV.4.4 *Both as view (BAV)*

Cette approche, proposée dans le cadre du projet AutoMed[70][71] par McBrien et Poulovassilis, consiste à *définir les correspondances comme une séquence de primitives de transformation*. Ces primitives sont appliquées pour obtenir les relations du schéma global ainsi que les données qu'elles contiennent. *La correspondance entre le schéma global et les sources de données peut se faire dans les deux sens, c'est-à-dire suivant la logique LAV ou GAV*. Il est en effet possible d'extraire le schéma global comme des vues sur les sources de données mais l'inverse est aussi vrai. Il est aussi possible d'extraire les sources de données comme des vues du schéma global. Nous allons, dans l'exemple suivant, rapidement décrire le principe du BAV. Dans cet exemple G est le schéma global que nous souhaitons construire et S_1 et S_2 sont les sources de données.

$$G$$
$$Auteur(id, nom, prenom, titre\_article)$$

$$S_1$$
$$Professeur(idp, nom, prenom, date\_naissance)$$

$$S_2$$
$$Article(ida, titre, conference, date, nom\_auteur)$$

Figure 16: Schéma global et sources BAV.

La définition des correspondances en BAV se fait à l'aide des primitives suivantes :

$$addRel(\langle\langle R, k_n \rangle\rangle, q)$$
$$addAtt(\langle\langle R, a \rangle\rangle, c, q)$$
$$delRel(\langle\langle R, k_n \rangle\rangle, q)$$
$$delAtt(\langle\langle R, a \rangle\rangle, c, q)$$

Figure 17: Primitive BAV.

La primitive $addRel(\langle\langle R, k_n \rangle\rangle, q)$ permet d'ajouter au schéma global une relation R avec

comme clés primaires $k_1 \ldots k_n, n \geq 1$, q permet de spécifier, en fonction des sources de données existantes, une requête définissant les valeurs que peut prendre k. La primitive addAtt permet de rajouter un attribut non clé, ayant une cardinalité c et dont les valeurs sont spécifiées par la requête q. Les primitives delRel et delAtt permettent de supprimer des relations ou des attributs respectivement.

Les expressions BAV permettant la construction du schéma global à partir des sources sont les suivantes :

$$addRel(\langle\langle Auteur, id \rangle\rangle, \{x | x \in \langle\langle S_{1,} idp \rangle\rangle \vee \langle\langle S_{2,} ida \rangle\rangle\})$$
$$addAtt(\langle\langle Auteur, nom \rangle\rangle, notnull, \{x, y | \langle x, y \rangle \in \langle\langle S_{1,} nom \rangle\rangle \wedge x \notin \langle S_{2,} ida \rangle \vee \langle\langle S_{2,} nom\_auteur \rangle\rangle\})$$
$$addAtt(\langle\langle Auteur, prenom \rangle\rangle, notnull, \{x, y | \langle x, y \rangle \in \langle\langle S_{1,} prenom \rangle\rangle\})$$
$$addAtt(\langle\langle Auteur, titre\_article \rangle\rangle, notnull, \{x, y | \langle x, y \rangle \in \langle\langle S_{2,} titre \rangle\rangle\})$$

Figure 18: Construction d'un schéma BAV.

À partir de ces règles de constructions, l'expression BAV pour la relation Auteur est la suivante :

$$Auteur(id, nom, prenom, titre\_article) = \{x, y_{1,} y_{2,} y_3 | (x \in \langle\langle S_{1,} idp \rangle\rangle \vee x \in \langle\langle S_{2,} ida \rangle\rangle)$$
$$\wedge (\langle x, y_1 \rangle \in \langle\langle S_{1,} nom \rangle\rangle \wedge x \notin \langle S_{2,} ida \rangle \vee \langle x, y_1 \rangle \in \langle\langle S_{2,} nom\_auteur \rangle\rangle)$$
$$\wedge (\langle x, y_2 \rangle \in \langle\langle S_{1,} prenom \rangle\rangle)$$
$$\wedge (\langle x, y_3 \rangle \in \langle\langle S_{2,} titre \rangle\rangle)\}$$

Figure 19: Correspondances BAV.

À partir de cette expression, il est possible d'utiliser d'autres primitives que celles mentionnées pour obtenir une expression de type GAV ou LAV. Le processus de transformation est détaillé dans [72]. Le BAV peut être considéré comme la plus flexible des approches proposées pour l'intégration de données. Il reste cependant coûteux à mettre en place dans le cas de nombreuses sources de données. En effet, pour tout changement du schéma global il sera nécessaire d'appliquer les primitives de transformation à tous les schémas conformes.

IV.4.5 *Synthèse*

Dans les sections précédentes, nous avons vu les différentes solutions pour l'intégration de données. L'approche *GAV présente l'inconvénient de rendre toute évolution des sources de données difficile*. Dans le cas de l'approche *LAV c'est l'évolution du schéma global qui est*

difficile. Dans un contexte où le domaine métier évolue rapidement et où chaque entreprise, dispose d'une grande autonomie et adapte ses systèmes d'information à son contexte, les sources de données ainsi que le schéma global sont amenés à évoluer. Les approches GLAV et BAV ont été proposées pour rendre plus flexibles les approches LAV et GAV. Le GLAV est une extension du LAV et permet, par l'intermédiaire de transformations complexes, la transformation d'un schéma LAV en GAV. Le BAV permet de construire des expressions sur lesquelles il est possible d'appliquer des primitives de transformation pour obtenir un schéma LAV ou GAV. *Le GLAV n'est qu'une extension du LAV et souffre, par conséquent, des mêmes inconvénients de ce dernier*. En effet, des relations permettent de transformer des expressions LAV en expressions GAV. Le BAV définit un ensemble de primitives permettant de construire un schéma global et de convertir ce dernier en un schéma GAV ou LAV afin de pouvoir bénéficier des avantages des deux méthodes de modélisation. L'inconvénient principal du *BAV est qu'il est beaucoup plus coûteux à mettre en place*. Dans la section suivante, nous verrons les différents impacts qu'ont les changements sur les correspondances.

IV.5 Impacts du changement

Dans le chapitre précédent, nous avons décrit les différentes méthodes de définition d'un schéma global. Dans celui-ci, nous avons montré les différentes méthodes pour relier un schéma global et les sources de données. Dans cette section, nous allons voir les différents, types de changement qui peuvent causer des modifications dans les sources de données et dans le schéma global. Ces modifications sont susceptibles de rendre le schéma global incohérent avec les sources de données ou inversement ce qui implique une incohérence des correspondances comme l'illustre la Figure 20.

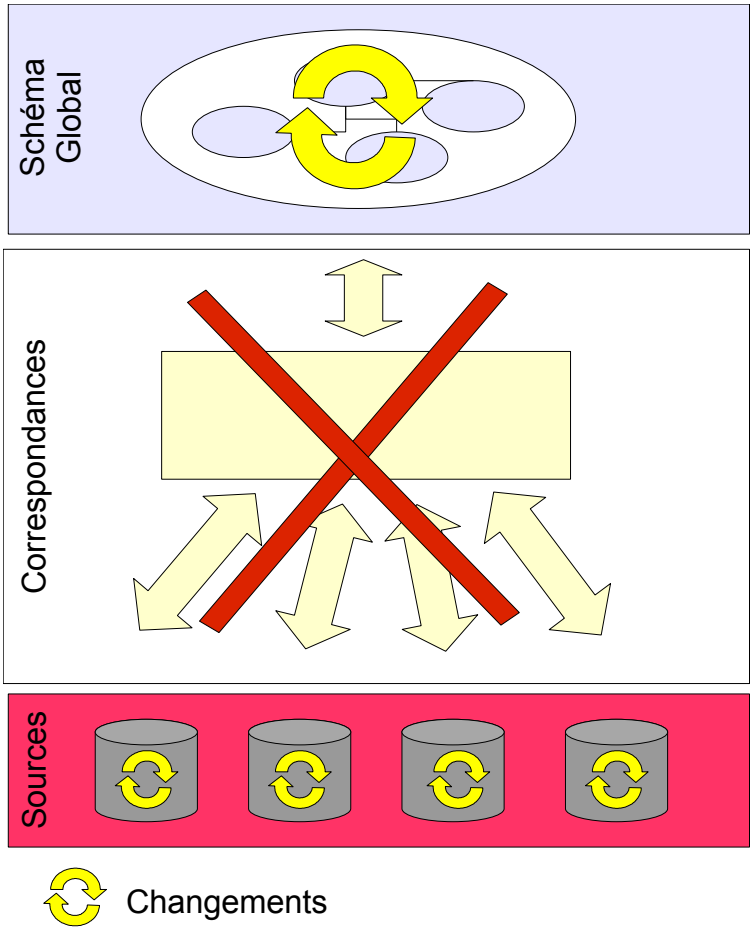

Figure 20: Impact des changements

IV.5.1 *Au niveau des sources de données*

Nous le rappelons, nous ne nous intéressons qu'aux changements de structure au niveau des sources de données. Ces changements de structures peuvent prendre la forme d'ajout, suppression ou modifications des relations et des attributs présents dans les sources de données.

Dans le cas *d'ajout de relations ou d'attributs, les correspondances ne sont pas affectées*. Il appartiendra à l'administrateur de décider si ces nouveaux éléments doivent être pris en compte dans l'intégration de données. Il définira par la suite de nouvelles correspondances afin de relier ces nouveaux éléments au schéma global.

Dans le cas de **suppression ou de renommage d'attributs**, *les correspondances peuvent être affectées*. En effet, dans une approche de type **GAV**, une telle modification fait que la *définition de la vue est inconsistante* avec ce qui est réellement présent dans les sources de données. Dans le cas de correspondances exprimées suivant une approche **LAV**, cette modification posera problème lors de la récupération des données, si elles ne sont pas matérialisées, *dans la mesure où les données demandées seront référencées par une autre étiquette ou une étiquette supprimée*.

Dans le cas de **suppression de relations**, les données contenues dans la relation sont aussi supprimées. *Les correspondances possédant des vues basées sur la relation supprimée en tête (LAV) ou faisant partie du corps (GAV) sont partiellement valides*. En effet, la description faite des données contenues dans la vue est erronée car elle décrit des données qui ne pourront être que partiellement récupérées. Ce problème se pose si les données ne sont pas matérialisées.

Dans le cas du **renommage ainsi que la division de relations**, le problème posé est *similaire à la suppression de relations sauf que les données sont présentes, mais référencées par une autre étiquette.*

Dans le cas de la **fusion de relations**, elle *invalide la définition attribuée aux correspondances*, car les relations sont inexistantes.

IV.5.2 *Au niveau du schéma global*

Au niveau des changements au sein du schéma global, *toute modification qu'elle soit le renommage, la suppression de concepts ou de rôles nécessite une mise à jour du corps des correspondances dans le cas d'une approche LAV et des entêtes pour une approche GAV*. Il est bien évident que la mise à jour dans une approche LAV est plus fastidieuse et risque de rendre incohérentes certaines correspondances. En effet, étant donné que les sources de données sont décrites en fonction des concepts du schéma global, la suppression d'un concept engendre la suppression d'une partie d'une correspondance ou sa suppression complète. Ceci peut rendre la recherche de reformulation impossible, car certaines concepts sont utilisés pour faire la liaison entre les variables d'autres correspondances.

Les approches hybrides ont été proposées pour justement pouvoir prendre en compte les changements au niveau du schéma global et au niveau des sources de données. Dans le cas du

GLAV, le traitement de changement n'est pas considéré. Dans le cas du *BAV, les changements sont traités en appliquant des primitives* du même ordre que celles décrites dans la section III.2.3.2. Une explication détaillée peut être lue dans [72].

IV.5.3 *Synthèse*

Nous avons vu, dans cette section, que **les changements** dans l'univers modélisé peuvent être issus de différents facteurs. Ils peuvent être liés à *l'évolution de l'univers*, il est rare que les choses soient statiques, mais aussi à l'*évolution de notre perception* des choses du fait d'une meilleure connaissance de l'univers. Ils peuvent aussi être dus à une volonté de *corriger certaines erreurs* de modélisation ou une *amélioration de la modélisation* afin de mieux refléter la réalité. Ces changements ne sont pas sans conséquence sur un schéma global ou sur les sources de données. Ils impliquent une évolution de contenu ou de structure. L'**évolution de contenu** dans le cas de sources de données provoque un besoin de mise à jour des vues matérialisées et n'a *pas d'impacts significatifs au niveau du schéma global*. En effet, les données sont rajoutées aux structures des sources de données, et ces structures sont déjà prises en compte dans le schéma global. Les **évolutions de structure**, quant à elles, ont plus d'incidences sur le SID. *Les correspondances doivent être mises à jour* afin de refléter l'ajout ou la suppression de relations, d'attributs ou de concepts, de rôles au niveau des sources de données et du schéma global respectivement. Les premières approches (GAV, LAV) permettant de définir les correspondances sont sensibles au changement de structures. Les approches hybrides ont été conçues pour justement prendre en compte ces changements, mais cette prise en compte n'est explicite que pour le BAV. Cependant ces solutions sont peu flexibles et la gestion du changement lourde à mettre en œuvre.

Chapitre V.

Architectures d'intégration

V.1 Introduction

Un schéma global et des correspondances permettent de décrire les connaissances abstraites, d'un ou plusieurs domaines, et les relations qui existent entre les connaissances décrites et les sources de données. Nous avons décrit dans le chapitre 2 les deux principaux modes d'organisation d'un SID à savoir le médiateur et le PDMS. Nous nous intéressons dans ce chapitre aux fonctionnalités qui nous paraissent essentielles pour un SID et pour la prise en compte du changement. Comme nous l'avons déjà spécifié dans les chapitres précédents, un SID synthétise le contenu de plusieurs sources de données suivant un besoin informationnel. Nous avons vu que pour faciliter son usage, le schéma global doit être exprimé suivant une terminologie métier spécifiant la sémantique des données contenues dans les différentes sources. De plus, afin de permettre la récupération des données, les utilisateurs doivent pouvoir exprimer et exécuter des requêtes afin d'obtenir les données référencées. Finalement, pour permettre la prise en compte des changements, le système doit pouvoir les détecter et les gérer.

Dans ce chapitre, nous étudions différents SID. Des nombreux SID existant dans la littérature, nous avons retenu ceux qui proposent des solutions par rapport à notre problématique. Nous analysons ces systèmes suivant les choix technologiques en termes de langage du schéma

global, de type de correspondances et d'architecture. Nous les analysons aussi suivant trois aspects :

- la présence de mécanismes de détection et de gestion du changement,
- la prise en compte de l'utilisateur, et, plus particulièrement la possibilité de modéliser les connaissances métiers,
- le mécanisme utilisé pour récupérer les données.

Ces trois grandes caractéristiques peuvent se décomposer en plusieurs critères. En effet, les critères qui nous intéressent pour la **détection et la gestion du changement** sont la *détection du changement au niveau des sources de données et du schéma global mais aussi la mise à jour des correspondances établies entre le schéma global et les sources de données. Cette détection est importante car elle permet d'identifier les éléments qui ont changés. Une fois identifié il est ensuite possible d'identifier les correspondances impactées et de les mettre à jour.*

Les critères d'analyse correspondant à la **prise en compte de l'utilisateur** consistent à identifier, d'une part *la présence d'un schéma global exprimé en fonction des connaissances métiers, et, d'autre part, la présence d'un mécanisme de reformulation de requêtes* en cas d'indisponibilité des sources de données.

Finalement les critères d'analyse pour la **récupération des données** sont la *validation syntaxique et la validation du contenu des requêtes avant la récupération effective des données*. La validation est importante car sans elle la récupération de données est impossible en cas de changements au niveau des sources de données.

V.2 INFOSLEUTH

Infosleuth est un projet du Microelectonics and Computer Technology Corporation (MCC) pour la collecte et l'analyse d'informations distribuées au sein de sources d'information autonomes. L'architecture d'Infosleuth est une architecture *médiateur* basée sur l'utilisation d'un SMA. Afin d'intégrer les informations des différentes sources de données Infosleuth utilise plusieurs ontologies de domaine formulées en OKBC [73][74]. Infosleuth suit une

approche *GAV* dans la mesure où chaque agent est responsable d'une source de données et doit publier le contenu de sa source de données en fonction des éléments qui se trouvent dans les ontologies de domaine.

Infosleuth, est un des premiers systèmes à introduire la notion de changement au sein des sources de données. En effet, les agents ressources qui encapsulent les sources de données sont responsables de la détection des changements au sein de leur source de données respective et de modifier si nécessaire la publication de leur contenu. Toutefois **aucune mise à jour des correspondances** n'est effectuée et les changements au sein du schéma global sont ignorés.

Ces ontologies permettent aux utilisateurs de savoir ce que contiennent des sources de données et peuvent ainsi formuler des requêtes pour récupérer des informations ou demander à être notifiés lorsqu'une information particulière est introduite dans une des sources de données. Cependant, ces **ontologies ne sont pas exprimées en fonction de connaissances métiers et aucune validation n'est effectuée lors de la récupération de données**.

V.3 AutoMed

Automed est une approche *BAV* qui utilise le modèle relationnel pour exprimer son schéma global. Cette approche peut être structurée pour former une architecture *médiateur* ou pair-à-pair[75]. La construction du schéma global dans AutoMed se fait en plusieurs étapes [72]. La première étape consiste à définir un vocabulaire commun à toutes les sources de données participant à l'intégration. Ce vocabulaire défini, la deuxième étape consiste à créer un schéma relationnel commun, appelé schéma conforme, à partir du vocabulaire défini. Les données de chacune des sources de données sont matérialisées dans une nouvelle base de données, appelée base de données conforme. Il existe autant de bases de données conformes qu'il y a de sources de données. À partir de ces schémas conformes, un schéma universel est construit en faisant l'union des schémas conformes. Ce schéma est un schéma virtuel dans la mesure où aucune donnée n'y est matérialisée. Le schéma universel permet, par la suite, de construire une source de données globale contenant les données des différentes sources. Ce **schéma universel obtenu n'est pas exprimé en fonction des connaissances métiers**. Contrairement

à Infosleuth qui ne prend en compte que les changements au niveau des sources de données, Automed permet aussi la prise en compte des évolutions du schéma global [72]. Cependant, **la prise en compte des changements au niveau des sources ou du schéma global est très coûteuse en termes de calcul**. En effet, il est nécessaire de reconstruire les sources de données conformes et la base universelle. Finalement **aucune validation n'est effectuée lors de la récupération des données**.

V.4 Semantic Webs and AgentS in Integrated Economies (SEWASIE)

SEWASIE[76][77][78] est un projet européen financé par l'Information Society Technologies. Ce projet a pour objectif de concevoir et implémenter un moteur de recherche avancé permettant un accès intelligent aux sources de données hétérogènes présentes sur le Web. SEWASIE suit une approche virtuelle de type *GAV* basée sur une architecture de *médiateurs*. En effet, les sources de données sont organisées en nœuds informationnels appelés SINodes. Ces noeuds peuvent regrouper plusieurs sources de données structurées ou semi-structurées distribuées. Ils sont publiés auprès d'un ou plusieurs agents. Ces derniers vont intégrer le schéma des différents SINodes à leur ontologie, toujours en suivant une approche GAV, après avoir effectué des tests de consistance. Les mécanismes d'extraction et d'intégration de données à partir de source de données structurées et semi-structurées de SEWASIE sont basés sur ceux élaborés dans le cadre du projet MOMIS.

SEWASIE ne propose **pas de mécanismes de détection et de gestion du changement** mais propose une approche originale pour l'extraction et la représentation du schéma global. Le mécanisme de construction de la représentation des connaissances est emprunté au projet MOMIS, décrit ci-dessous. En plus de ce mécanisme de construction, un système multi-agents est utilisé comme moyen d'exploitation des ontologies créées. Les agents permettent une récupération personnalisée des ontologies du système. A chaque requête utilisateur, les agents vérifient syntaxiquement la requête mais **aucune vérification du contenu de la requête n'est effectuée** avant de récupérer les données. SEWASIE peut également s'adapter à son environnement à savoir l'état du réseau et aux charges de traitement des serveurs. Les agents

du système peuvent migrer sur d'autres serveurs si les temps de calcul sont trop élevés. Ce mécanisme permet de traiter les requêtes utilisateur le plus rapidement possible.

V.5 SomeWhere

SomeWhere[79][80][81] est un SID *pair-à-pair*, où chaque pair possède un schéma défini en OWL-PL, un sous-langage d'OWL-DL. OWL-PL, PL pour logique de proposition, borne l'utilisation d'OWL-DL aux constructeurs de disjonction, conjonction et de négation.
La prise en compte de l'utilisateur se fait par l'intermédiaire d'ontologies composées d'un ensemble de descriptions de classes atomiques possédant chacune un nom unique, de relations d'équivalence, d'inclusion et de disjonction. L'ensemble des noms affectés aux classes atomiques d'un pair compose le vocabulaire de ce dernier. L'extension d'une ontologie est définie par des classes atomiques appelées classes extensionnelles. Ces classes extensionnelles sont des relations d'inclusion avec les classes atomiques dont elles définissent l'extension.
Les correspondances entre pairs sont définies manuellement par des relations d'inclusion, de disjonction et d'équivalence entre les concepts atomiques des pairs du système. Il est possible, à partir de ces correspondances, de définir un graphe d'accointances permettant de voir les relations existantes entre les pairs. Le graphe d'accointances est composé de noeuds, représentant les différents pairs, et d'arc entre les noeuds représentant les relations sémantiques entre les pairs. Il existe un arc entre deux pairs si et seulement si ils partagent une partie de leurs vocabulaires.
Afin d'apporter des réponses à une requête, posée en termes de classes et sur un pair quelconque, SomeWhere utilise le raisonnement distribué sur les théories en logique propositionnelle. La requête posée est décomposée en formules propositionnelles à partir des identifieurs de classes. En utilisant les correspondances définies, il est possible d'identifier les pairs partageant des classes similaires. Les requêtes sont propagées dans le réseau par un algorithme d'échange de messages.
SomeWhere **ne possède aucun mécanisme de détection de changement et aucun mécanisme de validation de requête au niveau des sources de données**.

V.6 TSIMMIS

The Stanford-IBM Manager of Multiple Information Sources (TSIMMIS) [82] est une approche *médiateur* qui utilise OEM (cf. section II.3.2) pour la représentation du schéma global. La liaison avec les sources de données se fait suivant une approche *GAV*. TSIMMIS propose d'exporter, par l'intermédiaire d'un adaptateur, toute ou partie d'une source de données en OEM. La description OEM obtenue est utilisée comme schéma global. La récupération des données se fait grâce à OEM-QL. Les requêtes en OEM-QL sont envoyées aux médiateurs qui peuvent soit reformuler la requête sur leur schéma et l'envoyer à leur adaptateur respectif soit demander à d'autres médiateurs de récupérer les données. Les résultats sont récupérés sous la forme d'objets OEM. Ces derniers sont renvoyés au médiateur pour des traitements d'harmonisation de type ou d'étiquettes. L'intérêt de TSIMMIS réside dans le fait qu'il propose un générateur de plan qui permet de calculer, à partir des descriptions des possibilités de chaque source de données, la meilleure façon, parmi les différentes possibilités, de récupérer les données. L'inconvénient majeur de TSIMMIS réside dans le fait qu'**il ne décrit pas sémantiquement les données contenues dans les sources** de données encapsulées par les médiateurs. En effet, pour les utilisateurs de TSIMMIS la connaissance contenue dans les sources de données, encapsulées par un médiateur, n'est pas explicité en dans une terminologie métier. TSIMMIS **ne propose pas de mécanismes de détection et de gestion du changement ni de reformulation de requête**.

V.7 SIMS

SIMS [83][84] pour Services and Information Management for decision Systems utilise LOOM(cf. Section II.5.3) pour définir un modèle de domaine et des modèles de sources de données. SIMS suit une approche de type *LAV* pour définir les relations entres les modèles et les sources de données. Le *médiateur* de SIMS permet de traduire les schémas de sources de données en LOOM afin de les intégrer au modèle du domaine. La traduction des sources de données en LOOM se fait par l'intermédiaire du module LOOM Interface Module (LIM). LIM correspond au médiateur de SIMS dans la mesure où il permet d'effectuer le pont entre

LOOM et une base de données mais aussi permet pour une requête écrite en LOOM de générer des requêtes dans le langage de la base de données et de convertir les résultats obtenus en instances de classe LOOM. L'intégration entre modèle de domaine et modèle de source se fait en définissant des relations entre les classes des deux modèles. L'originalité de SIMS est de proposer un mécanisme de récupération de données basé sur la planification. En effet, pour une requête posée sur le modèle du domaine, SIMS sélectionne les sources de données pertinentes et ordonne les sous-requêtes en utilisant un outil de planification appelé Prodigy. Tout comme TSIMMIS, SIMS **ne propose aucune représentation métier du contenu des sources de données et aucun mécanisme de détection et de gestion de changements.**

V.8 OBSERVER

OBSERVER [85], utilise la même logique que SIMS. Le schéma global dans l'approche OBSERVER est défini en CLASSIC. Dans OBSERVER, il n'y a pas de schéma intermédiaire entre le schéma global et les sources de données. Les correspondances entre le schéma global et les sources de données sont exprimées en algèbre relationnelle étendue et suivent une approche *GAV*.

Dans OBSERVER, le terme *médiateur* n'est pas explicitement utilisé. Cependant, il existe une entité présentant les fonctionnalités d'un médiateur ainsi que d'autres fonctions plus complexes. OBSERVER est un des premiers systèmes d'intégration qui tente de prendre en compte l'utilisateur à travers un ensemble d'ontologies décrites en logique de description. Il intègre aussi un mécanisme de raisonnement sur les ontologies, un catalogue des sources de données, des correspondances, un ensemble d'adaptateurs et une base de données auxiliaire. Afin d'accéder aux données encapsulées par un médiateur, l'utilisateur peut naviguer dans une ontologie qu'il aura sélectionnée pour découvrir le contenu des sources de données. Il peut ensuite formuler des requêtes en logique de description. Ces requêtes sont traduites en fonction des correspondances établies pour obtenir une expression en algèbre relationnelle étendue. Cette expression est ensuite décomposée en sous expressions permettant d'interroger les différentes sources de données et chaque sous-expression est traduite dans un langage

compréhensible par la source de données. Le résultat de chaque sous-expression est matérialisé dans la base de données auxiliaire. La requête de l'utilisateur est exécutée sur la base de données auxiliaire afin de fournir un résultat global à l'utilisateur. OBSERVER **ne propose pas de mécanismes de détection et de gestion du changement ni de reformulation de requête**.

V.9 MOMIS

Le SID MOMIS suit une approche *GAV*. L'aspect intéressant que propose MOMIS concerne la prise en compte de l'utilisateur par une construction semi-automatique des schéma globaux à partir des sources de données. MOMIS utilise un *médiateur* composé d'un générateur de schéma global et un gestionnaire de requête. Le générateur de schéma global permet à l'utilisateur de définir les sources de données qu'il souhaite intégrer. La construction de schémas globaux dans MOMIS suit 3 grandes étapes [86][87]:

- extraction des concepts présents dans les sources de données,
- annotation et construction d'un thésaurus,
- clusterisation des concepts et création de concepts globaux.

Une fois définies, les sources sont traduites en schémas ODLI3. Ces derniers sont annotés de manière semi-automatique en utilisant WordNet. Ces annotations permettent de construire un thésaurus commun aux différents schémas. Le thésaurus est utilisé pour le calcul d'indicateurs d'affinité de nommage et de structure permettant, par l'intermédiaire d'algorithmes de clusterisation hiérarchique, de regrouper les classes entre elles pour former une vue virtuelle globale. Cette vue peut ensuite être affinée par l'utilisateur du système. Le gestionnaire de requête utilise le processus standard de reformulation de requêtes, posées sur la vue virtuelle globale, à savoir la reformulation de la requête sur les sources de données concernées, d'optimisation de la requête, d'exécution, de fusion et de présentation des résultats. MOMIS propose une représentation métier mais **ne propose aucun mécanisme pour la détection des changements et la gestion des impacts sur la représentation définie.**

V.10 PICSEL

PICSEL[88] suit une approche *LAV* et utilise un *médiateur* permettant aux utilisateurs d'effectuer des recherches sur un schéma global et de récupérer les données correspondantes aux concepts du schéma global grâce à un langage de requête. La prise en compte de l'utilisateur se fait par la construction de schéma global. Le schéma est défini en CARIN-ALN et les concepts sont définis comme une hiérarchie et pour chaque concept un ensemble de propriétés est défini sous forme de rôles. Le schéma global est construit à partir de documents papier ou électroniques. Les administrateurs sont aussi mis à contribution lors de la construction d'une ontologie, dans la mesure où ils peuvent spécifier des contraintes ou rajouter des caractéristiques aux concepts définis.

En plus de la prise en compte des utilisateurs, PICSEL intègre aussi un mécanisme d'affinement de requête qui aide l'utilisateur à reformuler sa requête initiale si cette dernière ne renvoie pas de données. Cet affinement a lieu, soit lorsque l'utilisateur formule mal sa requête, soit lorsque la requête ne peut être mise en correspondance avec les sources d''informations. PICSEL ne prend pas en compte les situations où les informations supposées être dans les sources ne sont pas disponibles soit du fait de l'indisponibilité de la source de données elle-même ou suite à la modification de la structure de la source d'information. **Il ne dispose pas de mécanismes de gestion du changement.**

V.11 Synthèse

Dans ce chapitre, nous avons présenté différentes implémentations pour l'intégration de données. Elles utilisent les langages que nous avons décrits dans le chapitre III pour exprimer leur schéma global et suivent une des approches décrites dans le chapitre IV pour établir des correspondances entre le schéma global et les sources de données. Cependant, nous avons vu que ces éléments ne suffisent pas pour un SID. Il est nécessaire d'avoir un certain nombre de fonctionnalités afin de mieux adapter le système à l'utilisateur et afin de mieux gérer les changements qui influencent le système. Les fonctionnalités qui nous paraissent intéressantes sont :

- l'extraction automatique de concepts (MOMIS),
- l'utilisation d'ontologies pour expliciter la sémantique des connaissances décrites

(Infosleuth),

- la validation des requêtes utilisateur (SEWASIE),
- la gestion de changements (Infosleuth, Automed),

Le tableau présenté dans la Figure 21 résume les choix technologiques des approches existantes.

Approches	Architecture	Modèle de schéma global	Type de correspondances
TSIMMIS	Médiateur	OEM	GAV
OBSERVER	Médiateur	Logique/classic	GAV
Infosleuth	Médiateur/ Agent	Logique/OKBC	GAV
MOMIS	Médiateur/ Agent	ODLI3	GAV
SEWASIE	Médiateur/ Agent	ODLI3	GAV
SMS	Médiateur	Logique/LOOM	LAV
PICSEL	Médiateur	Logique/Carin	LAV
AutoMed	Médiateur	Relationnel	BAV
SomeWhere	Pair-à-pair	Logique/Owl	n/a

Figure 21: Synthèse technologique des approches existantes.

Les systèmes proposés ne traitent pas l'évolution des correspondances afin de prendre en compte les changements de structure. *Les deux seuls systèmes qui prennent en compte le changement sont Infosleuth et Automed*. Dans Infosleuth, tout changement au niveau des sources de données résulte en une nouvelle publication du contenu. Cependant, aucune précision n'a été apportée sur le mécanisme qui permet de relier les éléments modifiés aux concepts de l'ontologie permettant ainsi à l'*agent ressource* d'expliciter les concepts pour lesquels il ne peut plus fournir d'information. De plus, dans le contexte Infosleuth les changements du schéma global ne sont pas considérés.

Dans le cas de Automed, les auteurs affirment que les changements au niveau des sources de données ainsi qu'au niveau du schéma global sont traités. Cependant l'inconvénient principal est que le traitement de ces changements est déclenché manuellement. Il n'y a aucun mécanisme qui permet de détecter les changements et de déclencher les traitements appropriés.

La gestion du changement au sein d'un SID commence par sa détection. Comme nous l'avons vu dans le chapitre IV, les changements peuvent se produire au niveau des sources de données

mais aussi au niveau du schéma global. Afin de garder le SID cohérent, la mise à jour des correspondances est nécessaire. Le tableau de la Figure 22 résume les propositions des implémentations existantes en terme de gestion du changement.

Approches	détection du changement	Gestion du changement		mise à jour des correspondances
		Sources	Schéma	
TSIMMIS	non	non	non	non
OBSERVER	non	non	non	non
Infosleuth	non	oui	non	non
MOMIS	non	non	non	non
SEWASIE	non	non	non	non
SIMS	non	non	non	non
PICSEL	non	non	non	non
AutoMed	non	oui	oui	oui
SomeWhere	non	non	non	non

Figure 22: Critères d'analyse associés à la détection et la gestion du changement.

La prise en compte de l'utilisateur peut être décomposée en deux critères, l'expressivité du schéma global et la reformulation de requête. Nous pensons qu'un SID doit expliciter les données contenues dans les sources de données. Cependant *à l'exception de MOMIS, SEWASIE et SomeWhere, les systèmes existants se contentent de présenter les données ce qui donne des schémas globaux peu expressifs*. Les caractéristiques des systèmes existants concernant la prise en compte de l'utilisateur sont résumées dans la Figure 23.

Approches	Expressivité du schéma global	Reformulation de requête
TSIMMIS	peu expressif	non
OBSERVER	peu expressif	non
Infosleuth	peu expressif	non
MOMIS	expressif	non
SEWASIE	expressif	non
SIMS	peu expressif	non
PICSEL	peu expressif	non
AutoMed	peu expressif	non
SomeWhere	expressif	non

Figure 23: Critères d'analyse associés à la prise en compte de l'utilisateur.

La récupération des données peut être décomposée en validation syntaxique et validation du contenu de chaque requête adressée aux différentes sources de données. Un troisième critère est la récupération effective des données. Cependant, ce dernier critère est peu intéressant dans la mesure où il est rempli par toutes les solutions existantes. Nous ne pouvons pas en dire autant pour les deux premiers critères. En effet, les changements pouvant avoir lieu à tout moment, il est aussi nécessaire de valider les requêtes, non seulement syntaxiquement, mais il est aussi nécessaire de vérifier si les éléments à récupérer sont toujours présents dans leur source de données respective. Et, en cas d'invalidité, il serait intéressant de pouvoir proposer une requête alternative. *Seul MOMIS propose un mécanisme de validation syntaxique des requêtes mais aucune solution existante ne propose la validation de contenu des requêtes.* Ce mécanisme de vérification du contenu peut aussi être utilisé comme moyen de détection de changement. En effet, l'absence au sein d'une source de données d'un élément composant une requête implique que cet élément est présent dans le schéma global mais n'existe plus au niveau de la source de données. La Figure 24 résume les critères relatifs à la récupération des données pour les approches mentionnées.

Approches	Validation de requête		Récupération de données
	Syntaxique	Contenu	
TSIMMIS	non	non	oui
OBSERVER	non	non	oui
Infosleuth	non	non	oui
MOMIS	oui	non	oui
SEWASIE	non	non	oui
SIMS	non	non	oui
PICSEL	non	non	oui
AutoMed	non	non	oui
SomeWhere	non	non	oui

Figure 24: Critères d'analyse associés à la récupération de données.

V.12 Discussions

Les différentes solutions, présentées dans ce chapitre, sont intéressantes du point de vue des fonctionnalités offertes dans le cadre d'intégration. Cependant, de notre point de vue, *les différentes solutions ne traitent pas vraiment ou que partiellement le problème de l'évolution des sources de données et du schéma global*. En effet, nous avons vu que les sources de données pouvaient évoluer afin de mieux refléter la réalité, étendre les fonctionnalités offertes ou corriger des erreurs de modélisation. Ces changements, dans un contexte où les sources de données appartiennent à différentes entités ou services autonomes, peuvent se produire sans aucune notification par l'entité effectuant le changement. Les répercutions sur le SID peuvent être importantes, comme nous l'avons vu dans la section 2 du chapitre II. *Aucun système ne propose de mécanisme pour la détection des changements. La gestion du changement est, pour l'essentiel, faite manuellement.*

Nous pensons que la gestion des changements ne peut se faire sans un mécanisme de détection de changement au niveau des sources de données. La mise en place d'une entité

centrale pour la gestion du SID est problématique, et il est difficile de contraindre les entités participantes au SID à demander l'aval de cette entité avant toute modification. Le mécanisme de détection du changement est, par conséquent, utile pour l'identification du type de changement ainsi que des éléments ayant changés au sein des différentes entités participant au SID. Cette identification peut donner lieu à une évaluation de l'ampleur des mises à jour qui seront nécessaires pour prendre en compte les changements. En ce qui concerne les changements au niveau du schéma global, ils ne nécessitent pas de mécanismes de détection car aucun changement « automatique » ne peut se produire. En effet, les changements au niveau global sont exclusivement faits manuellement.

Nous avons, dans cette partie, vu les trois composants essentiels d'un SID. Le schéma global permet de décrire le contenu des sources de données de manière à ce que ce soit compréhensible par les utilisateurs. Un schéma global peut comme, nous l'avons dit, dans le chapitre III être modélisé sous forme d'un modèle relationnel, d'un modèle objet, d'un modèle XML ou d'un modèle exprimé en logique. Ce schéma global peut ensuite être relié aux sources de données en utilisant les approches GAV, LAV, GLAV et BAV. Afin de pouvoir prendre en compte l'évolution des sources de données et du schéma global, les approches GLAV et BAV sont privilégiées. L'architecture mise en place pour un SID peut être centralisée de type médiateur ou distribuée de type pair-à-pair.

À partir de notre contexte, *nous avons choisi de proposer une solution d'intégration basée sur une modélisation du contenu des sources de données en OWL-DL. Les correspondances suivent une logique BAV, dans la mesure où il est possible d'exprimer les sources de données en fonction des éléments du schéma global, mais l'inverse est aussi vrai. Ce type de correspondance nous permet de faire évoluer les sources de données ou le schéma global de manière indépendante. Le schéma global ainsi que les correspondances sont exploités par une architecture basée sur les SMA*. Nous souhaitons mieux exploiter les capacités des agents en matière d'adaptation et d'autonomie afin de rendre le SID plus flexible. Par flexibilité, nous souhaitons doter les agent de capacités pour la détection de changements au sein des sources de données, mais aussi de capacités leur permettant d'évaluer l'impact qu'ont ces changements au niveau de la représentation des connaissances. Etant donné que les entités (entreprise, département, service) participant au SID peuvent être autonomes, *nous utilisons l'autonomie des agents pour faciliter la détection des changements localement*. Cette détection peut donner lieu à une coopération avec les autres agents du système permettant d'évaluer

l'ampleur des mises à jour nécessaires. Mais elle peut aussi donner lieu à une *reformulation de la requête initiale afin de renvoyer un résultat ou une information sur l'état de la récupération des données à l'utilisateur*. Finalement, nous souhaitons pouvoir mettre à jour la représentation de connaissances afin de maintenir une consistance avec les sources de données.

Partie II

Intégration des données à partir des connaissances.

Nous avons vu en première partie les composants de base d'un SID. Il est, comme nous l'avons dit, composé d'un schéma global, de sources de données et de correspondances entre les sources de données et le schéma global. Nous avons successivement vu différentes formes de modélisation du schéma global, différentes méthodes de définition des correspondances ainsi que les architectures possibles. Cette deuxième partie est consacrée à notre proposition **KRISMAS** (Knowledge Representation Integration System with Multi Agent System) un système d'intégration à base de représentation des connaissances implanté à l'aide d'un système multi-agent.

Le système que nous proposons a, d'une part, pour objectif une **meilleure prise en compte des utilisateurs** à travers une représentation des connaissances métiers. D'autre part, KRISMAS permet aussi de **gérer les évolutions au sein des SID** à travers un mécanisme de mise à jour des correspondances offrant aux utilisateurs une garantie stable de bonne récupération des données. Nous verrons successivement la proposition en termes de modélisation du domaine métier et en termes d'architecture. La modélisation du domaine métier permet aux utilisateurs d'explorer le contenu des sources de données en fonction de leurs connaissances métiers et de récupérer les données associées malgré les changements qui peuvent être effectués au niveau des sources de données. Afin de prendre en compte les éventuels changements au niveau des sources de données participant aux systèmes

d'intégration, nous proposons une architecture qui nous permet de détecter les changements, évaluer leurs impacts au niveau des représentations de connaissances et, dans certains cas, mettre à jour ces derniers.

Nous décrivons en **chapitre VI** notre proposition en termes de *modélisation de connaissances métiers*. Nous montrons dans ce chapitre comment nous modélisons les connaissances métiers et comment nous les relions aux données. C'est dans cette façon de relier les connaissances aux données que réside notre principal apport. De plus, nous donnons les principaux algorithmes qui permettent aux utilisateurs de récupérer les données à partir des connaissances.

Les **chapitres VII et VIII** sont consacrés à l'*architecture de KRISMAS* basée sur des agents. Ces agents sont utilisés pour l'exploitation du schéma global et des correspondances définies dans le chapitre VI. Nous décrivons le fonctionnement des composants ainsi que leurs rôles.

Nous détaillons, dans le **chapitre IX**, les différents *protocoles mis en place pour la détection de changements, la récupération des données et l'ajout de nouvelles sources de données*.

Chapitre VI.

Modèle de connaissances

VI.1 Introduction

Au sein d'une entreprise, on distingue généralement deux types de connaissances : les connaissances tangibles et les connaissances intangibles. Les connaissances tangibles sont celles contenues dans les bases de données ou les documents appartenant à une entreprise. Les connaissances intangibles sont celles que possède le personnel de l'entreprise sous forme de compétences ou de savoir-faire, mais qui n'ont jamais été explicitées ou formalisées. Dans un environnement qui ne cesse d'évoluer, ces connaissances évoluent aussi ce qui est problématique si on base un SID sur une représentation des connaissances métiers. Il est donc nécessaire de pouvoir modéliser explicitement ces connaissances de manière à faciliter la prise en compte de leurs évolutions au sein du SID.

Comme nous l'avons vu en introduction de ce manuscrit, la masse de données générée par une entreprise devient problématique en termes de disponibilité, de consistance, d'utilisabilité, de sécurité. De plus, la multiplication des sources de données engendre des difficultés à identifier celles qui sont pertinentes pour la réalisation d'une tâche, mais aussi des difficultés à rapidement déterminer le contenu de ces sources de données. Il devient par conséquent difficile d'avoir un référentiel unique indiquant aux utilisateurs les données à privilégier lors de la réalisation de leurs tâches. Afin d'expliciter leur signification, les données sont

structurées ou organisées en informations[8]. Ces informations peuvent être transformées en connaissances[9] grâce à l'application d'un savoir-faire métier. Ce dernier peut prendre la forme de règles ou de contraintes qui sont appliquées pour prendre en compte les spécificités du métier. Le problème est que le savoir-faire métier n'est pas le même suivant les individus. En effet, chacun a sa propre interprétation d'une même information suivant l'expérience et le savoir dont il dispose.

Dans une société où la connaissance métier joue un rôle de plus en plus important dans les choix de stratégies de marché, dans l'optimisation des échanges intra ou inter entreprises et dans la définition de règles de bonnes pratiques pour atteindre de meilleures performances, il est important de partager et de mettre à la disposition du plus grand nombre ces connaissances métiers.

Afin de faciliter l'exploitation des données présentes au sein d'une entreprise, nous proposons de modéliser les connaissances fondamentales de différents métiers et d'expliciter les relations qui existent entre elles. Ces notions sont similaires à celles issues de la représentation des connaissances par objets [89]. À partir de cette modélisation, les utilisateurs pourront non seulement explorer et se familiariser avec les autres métiers de l'entreprise, mais pourront aussi récupérer les informations qu'ils souhaitent pour des besoins d'analyse ou de prise de décision.

Nous commencerons par décrire plus précisément ce que nous entendons par sources de données avant de voir comment modéliser les connaissances métiers. Une fois ces différents éléments décrits, nous définirons comment nous avons établi le lien entre les sources de données et la représentation métier. Finalement, nous donnerons les deux principaux algorithmes utilisés pour la récupération des données.

VI.2 Sources de données

Les sources de données sont des supports de stockage de données. Dans nos travaux, nous

8 Une information est un ensemble structuré de données. Nous pouvons par exemple dire qu'obtenir une information sur une personne consiste à connaître les données relatives à son nom, son prénom, son adresse.

9 Une connaissance correspond à un affinage d'une information suivant certaines règles. Si on reprend l'information sur la personne, nous pouvons dire que si celui-ci possède un diplôme en informatique alors il peut postuler sur un poste de concepteur développeur.

considérons qu'une source de données peut prendre la forme d'une source de données relationnelle ou XML. Nous pouvons découper ces sources de données en structures et attributs. *Une structure est un regroupement d'attributs* permettant de décrire un objet. *Les attributs représentent, quant à eux, les caractéristiques de l'objet métier*. Par exemple, une source de données démographique pourrait contenir des structures comme « Personne », « Logement » ou « Emploi » et la structure « Personne » pourrait contenir les caractéristiques de chaque individu telles que le nom, l'âge, la taille etc. Au sein d'une source de données relationnelle, la structure correspond aux tables et les attributs aux champs des tables. Dans le cas d'une source de données en XML, la structure serait un élément complexe (le parent immédiat des attributs) et les attributs des éléments simples.

Nous nous intéressons particulièrement aux attributs dans la mesure où ce sont eux qui référencent les données que nous souhaitons récupérer. *Nous définissons donc un attribut métier comme un quadruplet formé du nom de l'attribut, de son type, du nom de la source de données et du nom de la table si c'est une source de données relationnelle ou du nom de l'élément parent dans le cas d'une source XML.*

Pour le besoin des définitions, on désigne l'alphabet par Σ, les chaines de caractères par Σ^*, les types associés aux données par T et les valeurs que peut prendre une donnée par V.

Définition VI.2.A Attribut métier

Un attribut métier $\xi_i = (\chi, \gamma, \upsilon, \mu)$ $où \chi, \upsilon, \mu \in \Sigma^* et, \gamma \in T$ appartenant à l'ensemble des attributs métiers Ξ. χ correspond au nom de l'attribut (source de données relationnelle) ou un élément (source de données XML), υ le nom de la source de données, et μ le nom de la structure à laquelle il appartient. Ces deux derniers paramètres permettent d'identifier le lieu où se trouvent les données correspondant à l'attribut. γ correspond au type de l'attribut et appartient par conséquent à l'ensemble T des types de données.

Pour les besoins de la modélisation des connaissances métiers, il est nécessaire de pouvoir définir des contraintes sur les attributs métiers. Ces contraintes nous permettent non seulement de décrire les restrictions métiers qui sont applicables sur les attributs métiers, mais aussi de restreindre l'ensemble de données lors de sa récupération.

Définition VI.2.B Contrainte

En considérant que Op est l'ensemble des opérateurs de comparaison $Op=\{=,<,>,\leq,\geq,\neq\}$ l'ensemble des contraintes est définie sur $CT:\Xi\times Op\times V$. Une contrainte correspond donc à $ct_i=(\xi,op,v)$ tel que $v\in V$, qui représente une valeur, est conforme au type de l'attribut métier $\xi\in\Xi$ (cf Définition VI.2.A).

VI.3 Modélisation métier

Un **concept** est une *information sur les connaissances métiers*. Il regroupe les caractéristiques qui le décrivent. Un concept possède une étiquette $\sigma\in\Sigma^*$ permettant de spécifier la nature des connaissances décrites et un ensemble de propriétés $p_i\in P$ décrivant ses caractéristiques. Un concept peut s'apparenter à la notion de table dans le modèle relationnel.

Définition VI.3.A Concept

L'ensemble des concepts est défini sur $C:\Sigma^*\times 2^P$. Un concept $c\in C$ correspond à $c=(\sigma,\{p_1,\ldots,p_n\})$ tel que $\sigma\in\Sigma^*$ et $\{p_1,\ldots,p_n\}\in 2^P$.

Une **propriété** est l'élément de base de notre représentation et représente les *caractéristiques d'un concept*. elle possède une étiquette unique $\sigma\in\Sigma^*$ qui permet d'identifier ce qu'elle représente et un identifiant de correspondances $\psi\in\Psi$ où Ψ correspond à l'ensemble des correspondances (cf. définition VI.4.A) avec les sources de données du SID. *Une correspondance permet de spécifier l'emplacement des valeurs que peut prendre la propriété*. La notion de propriété est équivalente à la notion d'attribut dans le modèle relationnel ou de rôle en logique, de classe en OWL.

Définition VI.3.B Propriété d'un concept

L'ensemble des propriétés P est défini sur $P:\Sigma^*\times\Psi$. Une propriété correspond alors à $p=(\sigma,\psi)$ où $\psi\in\Psi$.

Une **relation** permet de spécifier *la nature du lien entre que deux concepts*. Par exemple, une

relation appelée « réalisation » peut être définie pour relier les concepts « technicien » et « tâche ». Cette relation permet de spécifier que c'est un technicien qui réalise une tâche (cf Figure 25). Il est possible d'avoir la relation inverse qui permet de dire qu'une tâche est réalisée par un technicien. En plus des deux concepts qu'elle relie, il est aussi possible de préciser les structures d'appartenance des attributs métiers (cf Définition VI.2.A) à utiliser pour la récupération des données. Ces structures sont exprimées sous la forme de chaînes de caractères.

Définition VI.3.C Relation

Soit l'ensemble des relations $R:\Sigma^* \times C \times C \times 2^{\Sigma^*}$ une relation possède une étiquette et trois paramètres qui représentent les concepts qui sont reliés ainsi qu'un ensemble de structures d'appartenance utilisé lors de la récupération et la fusion des données (cf. VI.5.2) $r=(\sigma, c_i, c_j, l) / c_i, c_j \in C, i \neq j, l \in \Sigma^*$

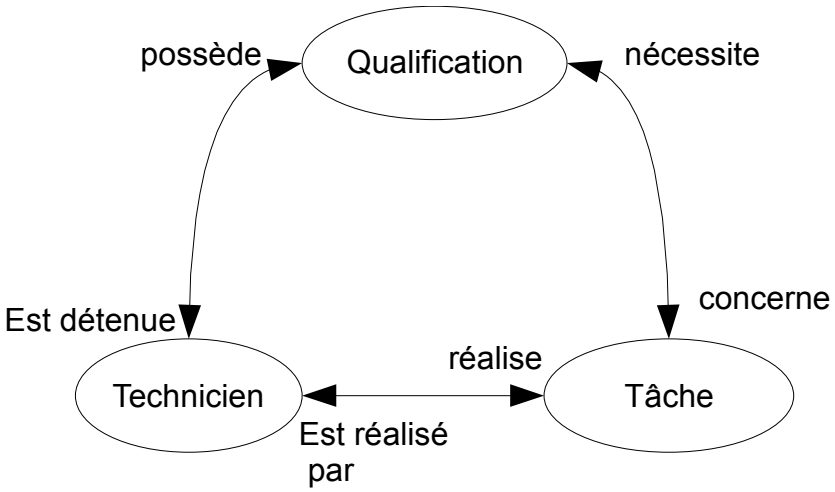

Figure 25: Les relations.

En résumé, les connaissances métiers sont définies en termes de concepts. *Un concept est composé d'un ensemble de propriétés et de relations. Pour chaque propriété, nous pouvons, définir un ensemble de correspondances spécifiant l'emplacement de leurs valeurs. Une relation décrit le lien entre deux concepts*.

VI.4 La correspondance avec les données

L'intérêt d'avoir un schéma global est non seulement d'expliciter les connaissances d'un métier, mais aussi de permettre la récupération des données à partir de ces connaissances. Nous avons spécifié qu'à chaque propriété nous associons un identifiant de correspondances. Cet identifiant permet de spécifier l'emplacement des données correspondant aux connaissances modélisées.

Nous avons défini, dans la section précédente, une propriété $p_i=(\sigma,\Psi)$, comme étant une étiquette associée à un identifiant de correspondances Ψ. *L'ordre au niveau des correspondances nous permet de décomposer* Ψ *en correspondance primaire et en correspondances secondaires*. La correspondance primaire définie le lien qui existe entre la propriété et la/les sources de données principales alors que les correspondances secondaires sont des liens avec une ou plusieurs sources de données secondaire.

Définition VI.4.A Ensemble des correspondances

L'ensemble des correspondances est un ensemble ordonné défini sur $\Psi:2^{\Theta}$. Un élément $\psi\in\Psi$ correspond à $\psi=(\theta_1,\ldots,\theta_n)/\theta_1\leqslant\theta_n$. Du fait de l'ordre défini sur Ψ nous pouvons considérer que le premier élément de l'ensemble est la correspondance primaire et les éléments suivants sont les correspondances secondaires.

Θ est une correspondance qui est aussi un ensemble ordonné d'attributs métiers associés à leurs contraintes. L'ordre est important puisque, par exemple, pour une adresse il est possible de la définir comme un triplet (rue, code postal, ville). L'absence d'ordre permettrait d'exprimer une adresse comme (code postal, ville, rue) ou (code postal, rue, ville) ce qui selon nous n'est pas équivalent. Θ est définie de la façon suivante $\Theta:2^{(\Xi\times 2^{CT})}$ tel que $\theta=\{(\xi_1,\{ct_1,\ldots,ct_n\}),\ldots,(\xi_n,\{ct_1,\ldots,ct_n\})\}$. Pour chaque élément de θ l'ensemble des contraintes définies s'applique à leur attribut métiers ξ_i respectif.

En plus des correspondances, il est nécessaire de définir des liaisons entre les différentes sources de données. En effet, *pour pouvoir intégrer des données issues de sources hétérogènes, les attributs métiers communs aux deux sources doivent être identifiés. Ces attributs métiers servent à définir des relations inter-sources* entre les différentes sources de

données. Les relations inter-sources sont utilisées lors de la fusion des données.

Définition VI.4.B Relations inter-sources

Les relations inter-sources sont des expressions d'égalité entre deux ensembles ordonnés d'attributs métiers issus de deux sources de données différentes. $\nu : \{\xi_1^1, \ldots, \xi_n^1\} = \{\xi_1^2, \ldots \xi_m^2\}$ tel que $\xi \in \Xi$.

Après avoir associé aux connaissances leurs emplacements, les utilisateurs peuvent explorer les connaissances, mais aussi récupérer les données associées en formulant des requêtes en fonction des connaissances disponibles. Nous décrivons dans la section suivante ce que nous entendons par requête et comment nous reformulons les requêtes exprimées en fonction des connaissances en requêtes compréhensibles par les sources de données.

VI.5 Des connaissances aux données

Afin de récupérer les données liées aux connaissances modélisées, les utilisateurs formulent des requêtes métiers qui forment l'ensemble Φ . *Une requête* $\phi = (\Pi, W)$ *où* $\phi \in \Phi$ *est formée de l'ensemble des propriétés que l'utilisateur souhaite récupérer* $\Pi, \Pi \neq \emptyset$ *et de l'ensemble de filtres* $W, W \neq \emptyset$ *applicables sur* Π . Les filtres permettent de restreindre l'ensemble des données renvoyées.

L'ensemble Π est composé de propriétés. Un utilisateur peut demander la récupération de données relatives à un ou plusieurs concepts, mais il peut aussi ne récupérer que des données relatives à certaines propriétés appartenant à un ou plusieurs concepts.

Définition VI.5.A Ensemble recherché

L'ensemble d'éléments sélectionnés est défini sur $\Pi : 2^P$. Si l'utilisateur veut sélectionner toutes les propriétés d'un concept, nous utilisons la fonction $propriétéDe(c)$ pour décomposer le concept en propriétés. La fonction est définie sur $propriétéDe : \Sigma^* \times 2^P$ tel que $propriétéDe(c) \rightarrow \{p_1, \ldots p_n\} / (c, 2^P) \in C$. Dans le cas contraire, la propriété est directement rajoutée à l'ensemble Π .

Pour chaque propriété sélectionnée, il est possible de spécifier des filtres. Les filtres permettent de restreindre les données renvoyées par les requêtes utilisateurs.

Définition VI.5.B Filtre

L'ensemble des filtres est défini sur $W : P \times Op \times V$. Un filtre $w_i = (p, op, v) / p \in \Pi$ est donc un triplet composé d'une propriété, d'un opérateur de comparaison et d'une valeur.

Afin de récupérer les données correspondant à la requête ϕ il est nécessaire de convertir ϕ en requêtes compréhensibles par les sources de données. Étant donné que la représentation des connaissances est formulée en termes de concepts métiers, il est nécessaire de les traduire dans un langage compréhensible par les sources de données. Comme dans les approches GAV, nous utilisons le remplacement des éléments composant une requête par leur définition.

VI.5.1 *Remplacement et regroupement*

Le principe général que nous proposons est de **remplacer les propriétés demandées par leur définition**. Afin de pouvoir remplacer les propriétés $p \in \Pi$ par les attributs métiers $\xi \in \Xi$ qui composent leur ensemble de correspondances $\psi \in \Psi$, nous utilisons les fonctions de "Manipulation des correspondances" définies ci-dessous, à savoir, $définitionDe : P \rightarrow \Psi$ qui nous permet de récupérer l'identifiant des correspondances pour une propriété p puis $sourcePrimaire : \Psi \rightarrow \Theta$ ou $sourceSecondaire : \Psi \rightarrow 2^{\Theta}$ pour la récupération des correspondances. Nous avons précisé que les correspondances étaient des ensembles ordonnés de couples d'attributs métiers et de contraintes. Le premier élément correspond à la source principale, celle qu'on privilégie pour la récupération de données. Dans l'éventualité où celle ci serait indisponible nous utilisons les sources secondaires pour la récupération de données. À partir de l'ensemble des correspondances Θ, nous utilisons les fonctions définies ci-dessous afin de récupérer les attributs métiers ξ de la correspondance primaire. Nous commençons par récupérer les correspondances avant de récupérer les attributs métiers qui les composent.

Fonctions VI.5.I Manipulation des correspondances

La fonction $définitionDe : P \rightarrow \Psi$ permet d'obtenir la définition ψ correspondant à une propriété p, $définitionDe(p) = \psi / \exists p = (\sigma, \psi)$. La récupération des sources se fait par l'intermédiaire de la fonction $sourcePrimaire : \Psi \rightarrow \Theta$ qui permet de récupérer la source principale associée à la définition d'une propriété: $sourcePrimaire(\psi) = \theta_1$ (θ_1 étant une correspondance) ou la fonction $sourceSecondaire : \Psi \rightarrow \Theta$ qui elle renvoie les sources secondaires. A partir de la source principale ou secondaire nous pouvons récupérer les attributs métiers grâce à la fonction $CorrespondanceDe : \Theta \rightarrow 2^{\Xi}$: $CorrespondanceDe(\theta) = \{\xi_, \ldots, \xi_n\}$. La fonction $contrainteSurAttribut : \Theta, \Xi \rightarrow Ct$ permet de récupérer la contrainte définie sur un attribut métiers: $contrainteSurAttribut(\theta, \xi) = \{ct_{1,}, \ldots, ct_n\} / (\xi, \{ct_{1,}, \ldots, ct_n\}) \in \theta$.

A partir de l'ensemble d'attributs métiers obtenu, nous les regroupons suivant leurs sources de données d'appartenance. *Cette appartenance est déterminée en récupérant, à partir des définitions, le nom de la source de données d'appartenance, les structures d'appartenance ainsi que les contraintes définies.* Les fonctions utilisées sont définies ci-dessous.

Fonctions VI.5.II Manipulation des attributs métiers

Les fonctions suivantes permettent de récupérer les différentes parties d'un attribut métier :

$AttributDe : \Xi \rightarrow \Sigma^*$ permet de récupérer le nom de l'attribut: $AttributDe(\xi) = \chi$,

$TypeDe : \Xi \rightarrow T$ permet d'obtenir le type de l'attribut: $TypeDe(\xi) = \tau$,

$SourceDeDonnées : \Xi \rightarrow \Sigma^*$ permet de récupérer le nom de la source de données: $SourceDeDonnées(\xi) = \upsilon$ et

$StructureDe : \Xi \rightarrow \Sigma^*$ permet d'obtenir le nom de la structure à laquelle l'attribut appartient: $StructureDe(\xi) = \mu$.

Ces informations sont utilisées pour **composer les différentes requêtes qui seront adressées aux différentes sources de données**. Ces différentes parties sont stockées dans une structure Λ, définie plus bas (cf Définition VI.5.C), qui est composée des attributs à récupérer, des structures contenant ces attributs, des contraintes qui sont applicables aux attributs et des

conditions de jointure qui permettent d'obtenir un résultat cohérent.

Définition VI.5.C Requête et ses fonctions

Soit $\Lambda : 2^{\Sigma^*} \times 2^{\Sigma^*} \times 2^{\Sigma^*} \times 2^{\Sigma^*}$ l'ensemble des requêtes, une requête λ correspond à $\lambda = (\{select_1, \ldots, select_i\}, \{from\_1, \ldots, from\_j\}, \{where_1, \ldots, where_k\}, \{join_1, \ldots, join_n\})$. Les fonctions suivantes permettent de récupérer respectivement:

- Les attributs qui permettent de répondre à la requête utilisateur. Ils sont récupérés grâce à la fonction $requêteSlt : \Lambda \rightarrow 2^{\Sigma^*}$: $requêteSlt(\lambda) = \{select_{1,} \ldots, select_i\}$
- la structure à laquelle appartiennent les attributs sélectionnés. Elle est renvoyée par la fonction $requêteSt : \Lambda \rightarrow 2^{\Sigma^*}$: $requêteSt(\lambda) = \{\text{from}_{1,} \ldots, \text{from}_j\}$
- les contraintes applicables sur chaque attribut. Elles sont obtenues grâce à la fonction $requêteCt : \Lambda \rightarrow 2^{\Sigma^*}$: $requêteCt(\lambda) = \{where_{1,} \ldots, where_k\}$
- les clauses de jointures entre les structures. Elles sont renvoyées par la fonction $requêteJ : \Lambda \rightarrow 2^{\Sigma^*}$: $requêteJ(\lambda) = \{join_{1,} \ldots, join_n\}$.

Afin de réaliser les opérations de remplacement et de regroupement, nous avons défini l'ensemble des *structures de stockage* Ω *qui va contenir, pour chaque source de données, l'ensemble des informations nécessaires à la composition de leurs requêtes respectives*.

Définition VI.5.D Ensemble de stockage et fonctions de manipulation

Soit $\Omega : 2^{(\Sigma^*, \Lambda)}$ un ensemble de stockage formé de couples composés du nom d'une source de données et d'une structure requête. Ω est donc composé de $\omega = \{(nomSource_{1,} \lambda_1), \ldots, (nomSource_n, \lambda_n)\}, n > 0$

Associées à l'ensemble de stockage nous avons défini deux fonctions $SourceInterrogée : \Omega \rightarrow 2^{\Sigma^*}$ et $RequêteSurSource : \Omega, \Sigma^* \rightarrow \Lambda$.

La fonction $SourceInterrogée$ permet de récupérer l'ensemble des sources de données qui sont concernées par une requête sur la représentation des connaissances $SourceInterrogée(\lambda) = \{nomSource_{1,} \ldots nomSource_n\}$.

La fonction $RequêteSurSource$ permet de récupérer pour une source de données la requête qui lui est adressée $RequêteSurSource(\omega, nomSource) = \lambda / (nomSource, \lambda) \in \Omega$

Ce processus de remplacement effectué, nous regroupons ensuite les fragments de la requête qui concernent la même source de données. Une fois les fragments regroupés nous déterminons les clauses de jointures, au sein de chaque source de données. *Les clauses de jointure sont déterminées à partir d'un sous-graphe reliant les différentes structures à partir desquelles seront extraites les données*. Le résultat final est une ou plusieurs requêtes pour chaque source de données permettant de récupérer les données.

Algorithme 1 : Remplacement et regroupement	
	Entrée : π , représentation des connaissances O , requête ϕ_o , ensemble de propriétés à reformuler $Ref = \{p_1, p_2, \dots p_n\}$
	Sortie : composants des requêtes organisés par sources de données
	Début
	$listeDesSources = \emptyset$ *// initialisation de la structure*
	Pour tout $p \in \Pi$ faire
	$\psi \leftarrow définitionDe(p)$ *// on récupère la définition pour une propriétés p*
	Si $p \in Ref$
	// si elle fait parti de la liste de propriétés à reformuler on récupère les sources secondaires sinon les sources primaires
	$\theta \leftarrow sourceSecondaire(\psi)$
	Sinon
	$\theta \leftarrow sourcePrimaire(\psi)$
	Fin si
	$attributs \leftarrow AttributDe(\theta)$ // on récupère les attributs de la définition
	Pour tout $a \in attributs$ faire
	$\xi \leftarrow correspondancesDe(a)$ // on récupère la correspondances de l'attribut
	$source \leftarrow sourceDeDonnées(\xi)$ // on récupère le nom de la source de données
	Si $source \notin sourceInterrogée(listeDesSources)$ alors
	// si la source ne fait pas partie des sources déjà traitées on initialise la structure $requête$ *et on l'ajoute à la liste des sources de données*
	$requête^{source} \leftarrow source$
	$listeDesSources \leftarrow listeDesSources \cup (source, requête)$
	Fin si
	Si $AttributDe(\xi) \notin requêteSlt(RequêteSurSource(listeDesSources, source))$ alors
	//si les attributs ne font pas partie des attributs déjà traités on l'ajoute à l'ensemble Slt *correspondant à la source de l'attribut. On ajoute aussi les contraintes associées à l'attribut à l'ensemble des contraintes*
	$requête_{Slt}^{source} \leftarrow requête_{Slt}^{source} \cup attributDe(\xi)$
	$contrainte \leftarrow contrainteSurAttribut(\theta, \xi)$
	Pour tout $c \in contrainte$
	// pour toutes les contraintes métiers définies sur un attribut on récupère l'opérateur de contrainte ainsi que la valeur de la contrainte avant de l'ajouter à l'ensemble des contraintes applicables sur les attributs d'une sources de données
	$requête_{Ct}^{source} \leftarrow requête_{Ct}^{source} \cup AttributDe(\xi).OperationDeContrainte(c).ValeurDeContrrainte(c)$
	fin pour

	fin si
	Si $structureDe(\xi) \notin requêteSt(listeDesSources, source)$ alors
	// si la structure de l'attribut n'est pas déjà présente dans l'ensemble St on l'ajoute
	$requête_{St}^{source} \leftarrow requête_{St}^{source} \cup structureDe(\xi)$
	Fin si
	Pour tout $w \in listFiltre(\phi)$ faire
	// on recherche les filtres définis sur la propriété p et on les ajoute à l'ensemble des contraintes
	si $proprietéFiltre(w) = p$ alors
	$requête_{Ct}^{source} \leftarrow AttributDe(\xi).OperationDeContrainte(w).ValeurDeContrainte(w) \cup requête_{Ct}^{source}$
	fin si
	Fin pour
	Fin pour
	Fin pour
	Fin

VI.5.2 *Définition des jointures*

Une fois les différentes parties de(s) requête(s) définies, l'étape suivante consiste à trouver les liens existants entre les éléments de l'ensemble des structures St de chaque requête $\lambda \in \omega$ pour chaque source de données. En effet, une requête peut concerner plusieurs tables ou éléments. Dans ce cas, relier ces structures permet de retourner un ensemble de données cohérent et satisfaisant la demande de l'utilisateur. La recherche de liens se fait au niveau de chaque source de données et permet de définir les clauses de jointure pour chaque requête.

Afin de définir la/les clause(s) de jointure, *nous définissons pour chaque source de données un graphe*. Dans le cas d'une **source de données XML**, le graphe prend la forme d'un *graphe non orienté* et dans le cas d'une **source de données relationnelle** il prend la forme d'un *graphe orienté acyclique*. En effet, l'absence de contraintes d'intégrité au sein d'une source de données XML explique la nature non-orienté du graphe. Le modèle relationnel permet, quant à lui, la définition contraintes d'intégrité par l'intermédiaire de clés primaires/clés étrangères interdisant de ce fait tout cycle.

Ces graphes nous permettent de trouver un ou plusieurs chemins reliant les tables au sein de chaque source de données. Parmi ces chemins, nous choisissons les chemins avec les coûts de jointure les moins importants afin de réduire le temps de récupération des données. Nous verrons dans les deux paragraphes suivants la construction des graphes pour les sources de

données relationnelles et XML.

Dans le cas d'une base de données, le graphe obtenu est un graphe orienté acyclique. Les noeuds représentent les tables de la base de données et les arêtes les relations clés primaires/clés étrangères existantes entre les tables. Les arêtes $\delta=(v, v', j, x), v \neq v', j \in \Sigma^*, x \in \mathbb{R}$ sont définies par deux valeurs, j représentant la condition de jointure et x une estimation du coût de récupération des données. La condition de jointure entre deux noeuds correspond aux attributs qui seront utilisés pour combiner le contenu des deux tables. L'estimation du coût de récupération des données se fait en prenant les tables deux à deux. Ce coût est obtenu en utilisant les outils statistiques fournis par les SGBD. Ce coût est maximal dans la mesure où on n'applique pas de critères de restriction.

Dans le cas de sources de données XML, les noeuds représentent les éléments et les arêtes une relation entre deux noeuds. La définition de δ est simplifiée, car il n'y a pas de paramètre pour représenter le coût des jointures. Les conditions de jointures sont spécifiées en utilisant des expressions en Xpath. Ces expressions permettent d'exprimer des équivalences entre différents éléments contenus dans un document XML. Une arête correspond dans ce cas à $\delta=(v, v', j), v \neq v', j \in A^*$.

Définition VI.5.E Construction du graphe

Nous rappelons que S est l'ensemble des sources de données et qu'une source est composée de schémas de relations $S=\{R_1, \ldots, R_2\}$ et que $R_k=(A_i : D_i \ldots A_n : D_n)$ où A_i correspond à un attribut prenant sa valeur dans D_i.

La fonction $buildGraph : S \rightarrow G$ permet à partir d'une source de données de construire un graphe composé de noeuds V et d'arêtes Δ tel que $G=(V, \Delta)$. Chaque noeud représente un schéma de relation appartenant à la source de données $V=\{v_1 \ldots v_n\}, n \leqslant |R|$ et une arête représente l'existence d'une contrainte d'intégrité référentielle entre deux schémas de relation. L'ensemble $\Delta=\{\delta_1, \ldots, \delta_2\}$ d'arêtes est un sous-ensemble du produit cartésien $V \times V$ et est composé d'arêtes $\delta=\{(v_m, v_t) \in V, m \neq t | \exists m, t \in \{1 \ldots |R|\}, (R_m, R_t)\}$. Pour une arête reliant les noeuds v_m et v_t il existe obligatoirement une relation clé primaire/clé étrangère entre les schémas de relation R_m et R_t.

Le principe de l'algorithme 2 « calcul des jointures » est de trouver un **sous-graphe partiel**

qui permet de relier l'ensemble des structures nécessaires, au sein de chaque source de données, pour répondre à la requête utilisateur. Nous utilisons, pour des raisons de simplification, la notation $requête_X^S$ pour désigner l'ensemble X appartenant à une *requête* sur une source S. *Le sous-graphe partiel est construit à partir des chemins qui relient les structures* $st \in requête_{St}^{source}$. Un chemin c est une suite de sommets $c = \{st_1, \ldots, st_n\}$ telle que $\forall i = \{1, \ldots, n-1\} (st_i, st_{i+1}) \in \Delta$. À partir de l'ensemble des chemins trouvés, nous prenons le chemin reliant le plus de noeuds, appelé le chemin maximal, et vérifions que les autres chemins ne sont pas inclus dans ce dernier. Si un chemin n'est pas inclus, nous vérifions l'existence d'un noeud commun avec le chemin maximal. En présence de noeuds communs, nous pouvons rajouter le chemin au chemin maximal. Dans le cas contraire, nous utilisons le graphe de la source de données pour déterminer les structures qui permettent d'accéder aux noeuds du chemin isolé. Ces structures sont rajoutées à l'ensemble $requête_{St}^{source}$ et la recherche de chemin est relancée jusqu'à trouver un sous-graphe connexe.

Une fois le sous-graphe construit nous déterminons le plus court chemin reliant les différentes structures. Afin de garantir que seuls les chemins nécessaires sont conservés, nous vérifions que le sous graphe partiel ne contient pas les autres chemins trouvés. Nous vérifions aussi que les chemins trouvés possèdent au moins un noeud figurant parmi les éléments sélectionnés et que les arêtes qu'il partage forme avec d'autres noeuds font partie d'une chaîne entre deux noeuds de St.

Par exemple, la Figure 26 représente le graphe d'une source de données s et les structures st. Supposons que, pour apporter une réponse à la requête utilisateur les structures nécessaires sont C, E et G, déterminées à partir des propriétés demandées par l'utilisateur. À partir du graphe de la source de données, nous recherchons tous les chemins avec comme point de départ ou d'arrivée C, E et G. Le résultat de la recherche est le chemin EFG qui permet de relier E et G mais C reste isolé comme illustré dans la Figure 27. Afin de trouver un chemin qui relie les trois structures, nous rajoutons les noeuds parents de E,C,G c'est à dire A, B et H comme le montre la Figure 28. Les chemins possibles entre les éléments de St sont recalculés et le résultat est ;

$$\{(A,G),(B,A),(B,E),(B,C),(B,E,F),(F,G),\\ (B,A,G),(B,E,F,G),(E,F),(E,F,G),(H,C)\}$$

Seulement deux sous-graphe sont conservés pour la jointure entre C,E et G. Ces sous-graphe

sont $\{(B,C),(B,E),(B,A,G)\}$ ou $\{(B,C),(B,E,F,G)\}$. Admettons que la deuxième solution est retenue parce qu'elle est moins coûteuse ou que l'administrateur a spécifié que ce chemin doit être utilisé pour relier les noeuds B, E, F, G. Dans ce cas le chemin $\{B,A,G\}$ faisant partie du sous-graphe est inutile. Il est supprimé parce qu'il ne fait pas partie de la chaîne retenue pour la définition de la clause de jointure. On élimine aussi tous les chemins dont les noeuds de départ et d'arrivée sont inclus dans le sous-graphe partiel sélectionné. Dans le cas de notre exemple $\{(A,G),(B,A),(B,E),(B,E,F),(B,A,G),(E,F),(E,F,G),(H,C)\}$ sont supprimés, car soit ils sont inclus dans le sous-graphe sélectionné, soit qu'ils sont inclus dans un chemin non retenu.

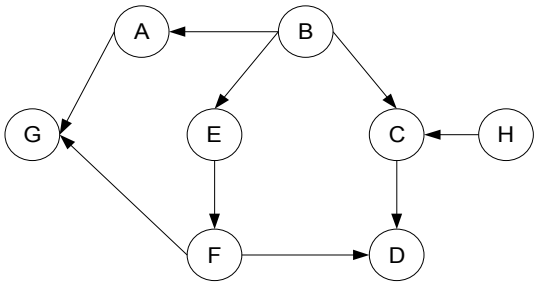

Figure 26: Graphe de la source s.

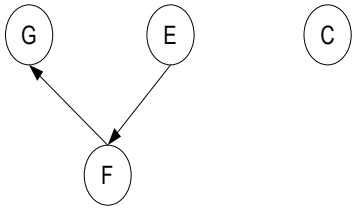

Figure 27: Nœud isolé.

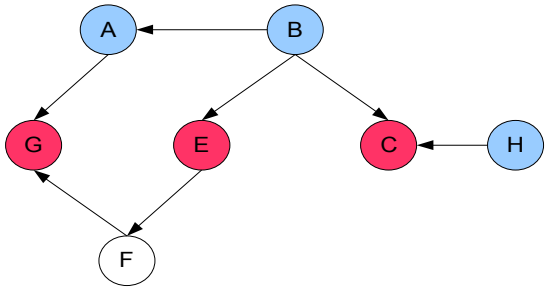

Figure 28: Sous-graphe d'une requête.

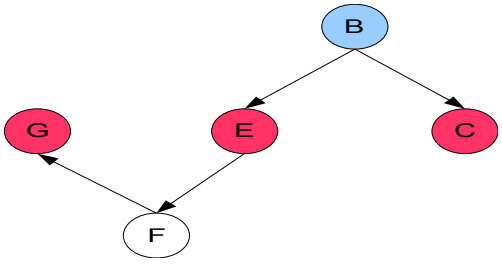

Figure 29: Graphe partiel de jointure.

Le résultat final, Figure 29, est utilisé pour déterminer les clauses de jointures. *Si un sous-graphe ne peut être trouvé et qu'il existe toujours un ou plusieurs chemins isolés alors plusieurs requêtes seront adressées à la même source de données*. Dans ce cas précis, il appartiendra à l'utilisateur de fusionner les ensembles renvoyés.

L'algorithme 2 utilise le résultat de l'algorithme de regroupement et de remplacement. En effet ce dernier nous permet de transformer une requête formulée sur la représentation des connaissances et requêtes compréhensibles par les sources de données. L'algorithme de calcul des jointures nous permet, pour chaque sources de données concernées, de définir les contraintes de jointure pour les structures composants $requêtes_{st}^{source}$. Ces contraintes de jointures nous permettent de récupérer une ensemble cohérent de données au niveau de chaque sources de données.

Algorithme 2 : Calcul des jointures	
	Entrée : $listeDesSources$, ensemble des graphes des sources de données G
	Sortie : clauses de jointure $requête_J$ pour chaque source de données
	Début
	Pour tout $source \in sourceInterrogée(listeDesSources)$ faire
	// On va rechercher pour chaque source de données interrogée un graphe connexe reliant les structures de chaque requêtes
	Tant que $connexe = faux$ ou $requête_{St}^{source} \subset V , V \in G$ faire
	// tant que le graphe n'est pas connexe ou que tous les nœuds n'ont pas été ajoutés on recherche les chemins
	Pour tout $st , st' \in requêteSurSource(source) , st \neq st' , st , st' \notin cheminPotentiel$
	// pour toutes les structures, pris deux à deux, faisant partie de l'ensemble st de chaque sources de données
	Si $\exists chemin(st , st' , G_{source}) / st \neq st'$ alors
	// s'il existe un chemin reliant les deux structures on ajoute le chemin aux chemins potentiels
	$cheminPotentiel_{source} \leftarrow cheminPotentiel_{source} \cup chemin(st , st' , G_{source})$
	Fin si
	Fin pour
	Pour tout $st , st' \in requête_{St}^{source}$ faire
	// on vérifie que tous les chemins sont joints si ce n'est pas le cas on rajoute les relations des noeuds
	Si $\nexists c \in cheminPotentiel$ joignant (st , st') ou (st' , st) alors
	$connexe \leftarrow faux$
	$requête_{St}^{source} \leftarrow requête_{St}^{source} \cup relationsDe(st) \cup relationsDe(st')$
	Fin si
	Fin pour
	Fin tant que
	Pour tout $c \in cheminPotentiel$ faire
	//on vérifie que les chemins trouvé ne font pas partie d'une chaine reliant deux autres structures si c'est le cas on le supprime.
	Si $st , st' \in c / st , st' \notin chaine(st'' , st''') / st'' , st''' \in requête_{St}^{source}$
	$supression(c , cheminPotentiel)$
	Fin si

	Fin pour
	$cheminFinal \leftarrow cheminMaximal(cheminPotentiel)$
	Pour tout $chemin \in cheminPotentiel_{source}$ faire
	// on vérifie que tous les chemins sont inclus dans dans le chemin maximal si ce n'est pas le cas on le rajoute à l'ensemble de chemins finaux.
	Si $chemin \not\subset cheminFinal$ alors
	$cheminFinal \leftarrow cheminFinal \cup chemin$
	Fin si
	Fin pour
	$requête_J^{source} \leftarrow cheminFinal$
	Fin pour
	fin

Maintenant que la requête utilisateur est convertie et que les différentes parties des requêtes adressées aux sources de données définies, nous pouvons procéder à la récupération des données. Cette étape est prise en charge par des agents ressources. Ces derniers font partie d'un système multi-agents que nous décrivons dans le chapitre suivant.

VI.6 Synthèse

Nous avons décrit dans ce chapitre notre proposition en termes de représentation de connaissances. Nous nous sommes attachés à décrire les connaissances métiers en fonction de concepts, propriétés et de relations. Une fois la représentation des connaissances définie nous identifions les sources de données qui contiennent les données relatives au modèle métier et nous les associons aux connaissances manuellement. En résumé, les connaissances métiers sont définies en terme de concepts qui sont un regroupement de propriétés. Les propriétés ont une étiquette qui définit leur nom et possède un ensemble de correspondances qui permettent de récupérer les données correspondantes à partir d'une ou plusieurs sources de données. Il est possible de définir pour chaque correspondance des contraintes qui agissent comme filtres afin d'obtenir des données pertinentes. Les concepts sont reliés entre eux et une relation entre deux concepts peut être sujette à la validité d'une contrainte sur les valeurs des propriétés participantes.

Nous verrons dans le chapitre suivant comment cette représentation des connaissances peut

être utilisée par un système multi-agents pour faciliter l'exploration du contenu des sources de données à partir de connaissances métier, mais aussi les différents protocoles mis en place pour la gestion des correspondances définies entre les sources de données et la représentation des connaissances. Nous aborderons aussi la question de l'évolution de la représentation des connaissances par l'ajout ou la suppression de propriétés, concepts, relations.

Chapitre VII.

Architecture Multi-Agent pour l'intégration de données.

VII.1 Introduction

Dans ce chapitre nous présentons l'architecture de KRISMAS, qui permet un *faible couplage* entre les applications et les sources de données du SID comme l'illustre la Figure 30. KRISMAS permet *d'isoler les utilisateurs et les applications des changements* au sein des sources de données tout en *exposant les données dans une terminologie métier*. Afin de remplir cette fonction d'isolant, il comporte un ensemble de fonctionnalités lui permet de *détecter et de prendre en compte les changements de structure* au niveau des sources de données tout en minimisant les impacts de ces changements sur le reste du système.

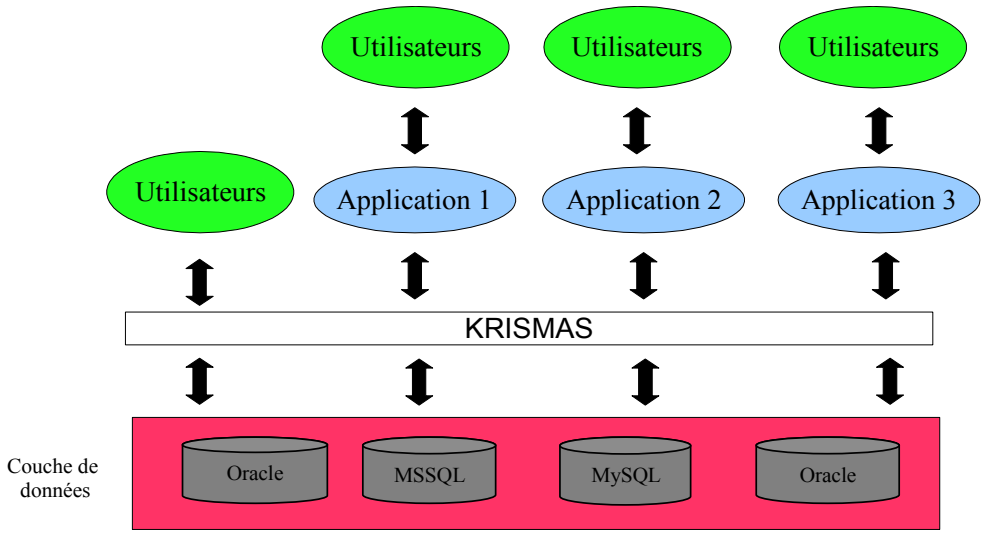

Figure 30: Le système KRISMAS au sein d'un SID.

Dans le chapitre précédent, nous avons détaillé notre proposition en matière de représentation des connaissances. Celle-ci est utilisée pour décrire un domaine métier et récupérer les données associées aux connaissances. Comme nous l'avons décrit au début de ce manuscrit, ces connaissances sont relatives à une entreprise qui est composée de plusieurs entités autonomes réparties géographiquement. L'autonomie dont disposent ces entités leur permet d'effectuer des changements dans la structure de leurs sources de données sans aucune notification aux autres entités. Ces changements, comme nous l'avons décrit dans le chapitre IV section 5, engendrent des ruptures de correspondances qui peuvent rendre le SID partiellement inexploitable. En effet, si les correspondances ne sont plus valables il est impossible de récupérer les données associées aux propriétés ou concepts décrivant les objets métiers.

Afin de détecter ces éventuelles ruptures et de les réparer, nous proposons d'utiliser un système multi-agents (SMA). Nous avons choisi de doter KRISMAS d'une architecture à base d'agents compte tenu des avantages en termes d'autonomie des agents, de communication et d'évolution du système. L'*autonomie nous permet d'utiliser les agents au niveau de chaque source de données pour la détection de changements*, la communication nous permet de prendre en compte l'environnement distribué. L'évolution du système concerne l'ajout ou la suppression de sources de données mais aussi les modifications de la structure d'une source de

données. L'*ajout d'une nouvelle source de données consisterait à déployer un nouvel agent* sans remettre en cause le système existant. Le *retrait d'une source de donnée engendre la suppression de l'agent ressource* et implique que les données relatives aux propriétés, qu'ont cette source définie comme source primaire ou secondaire, ne pourront être récupérées. Dans ce cas, l'utilisateur est averti qu'aucune réponse complète ne pourra être fournie.

Dans ce chapitre, nous allons commencer par décrire **les comportements des principaux agents** avant de voir le **fonctionnement général du système**. Nous détaillons le comportement des agents avec une attention particulière pour les agents responsables des sources de données et pour l'agent responsable de(s) ontologie(s) dans la mesure où ce sont les principaux acteurs de la détection du changement, de la récupération de données et de l'évolution du système.

VII.2 Architecture

Afin de faciliter l'utilisation de la représentation des connaissances par les utilisateurs et par des applications, nous introduisons un **SMA** satisfaisant les besoins de 1) *construction et interprétation de l'ontologie* et 2) *gestion de l'évolution des connaissances et maintien de la cohérence du système*.

Afin de réaliser ces objectifs, le SMA que nous proposons est composé :

- d'un module de gestion et de manipulation des ontologies,
- d'un module de veille des sources de données et de récupération des données,
- d'un module de services aux agents,
- d'un module d'administration.

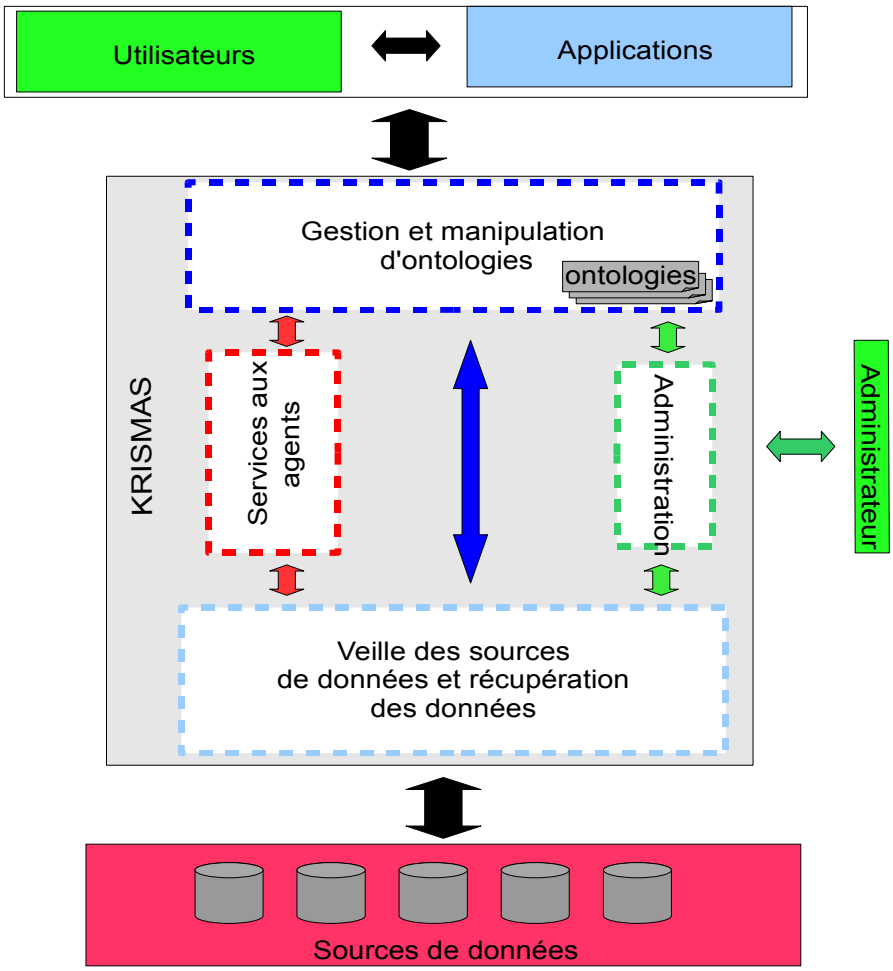

Figure 31: Modules de l'architecture KRISMAS.

Les différents modules de KRISMAS sont composés d'agents et les interactions entre les différents modules sont illustrées dans la Figure 31 ci-dessus. Les interactions prennent la forme de protocoles entre les agents qui composent les modules.

VII.3 Structure des agents et modélisation

Dans cette section nous nous attachons à définir l'organisation interne des agents qui composent KRISMAS et comment ceux-ci sont modélisés en AML (Agent Modeling Language) [90].

Un agent peut avoir plusieurs objectifs à réaliser. Chaque objectif peut nécessiter l'exécution d'une tâche ou d'un ensemble de tâches afin d'être réalisé. *Les agents sont définis par un*

ensemble de comportements en relation avec les objectifs à réaliser. Chaque comportement regroupe un ensemble de tâches à réaliser. L'exécution des tâches peut entraîner la coopération d'autres agents. La coopération entre agents se fait par échanges de messages suivant un protocole spécifique appelé protocole d'interaction.

Dans le reste de ce chapitre, les agents sont décrits par des classes au sens UML ayant le stéréotype <<Agent>> et sont représentés par l'icône et les comportements par des classes ayant le stéréotype << behaviour fragment>> et par l'icône . Les méthodes des classes de comportement représentent les tâches qui peuvent conduire à la réalisation de l'objectif.

VII.4 Gestion et manipulation d'ontologies

Le **module de gestion et manipulation d'ontologies** est composé d'un *agent ontologies* et d'un *ensemble d'ontologies métiers*. L'*agent ontologies* agit comme serveur d'ontologies. Afin de réaliser l'ensemble des tâches relatives à la gestion d'ontologies nous avons doté cet agent de différents comportements représentés dans la Figure 32:

- maintenance de l'ontologie,
- évaluation de l'impact des changements sur l'ontologie,
- conversion de requêtes et
- navigation au sein d'ontologies

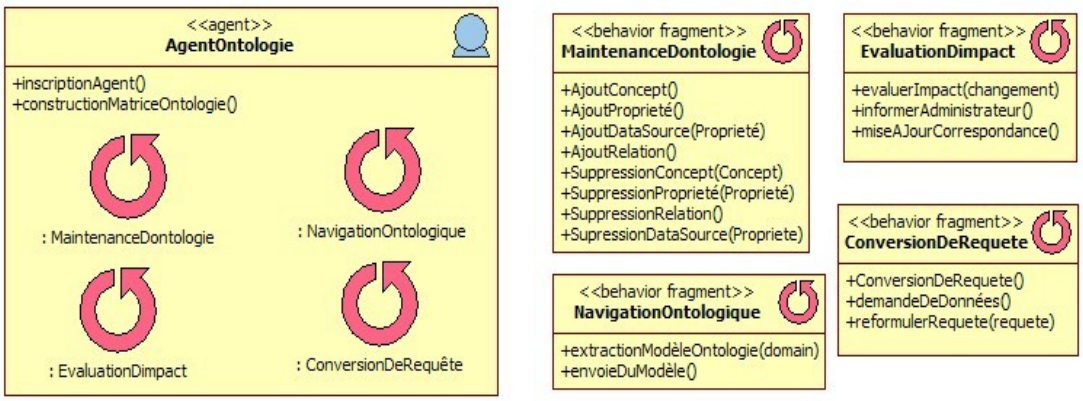

Figure 32: Comportements de l'agent ontologies.

Dans les sous-sections suivantes, nous décrivons le fonctionnement de chaque comportement.

VII.4.1 *Maintenance d'ontologies*

La **maintenance d'ontologies** concerne la gestion des modifications relatives à *l'ajout, la modification ou la suppression de concepts, propriétés, correspondances et sources de données*. Ces tâches de gestion d'ontologie sont *déclenchées par l'administrateur* du système et une fois effectuées, elles sont enregistrées afin de garder une trace des évolutions. Le comportement « MaintenanceDontologie » (cf Figure 32) regroupe l'ensemble des tâches suivantes :

- Les tâches AjoutConcept/SupressionConcept permettent l'ajout et la suppression d'un concept au niveau des ontologies que gère l'agent.
- Les tâches AjoutProperty/SupressionProperty permettent l'ajout et la suppression d'une propriété au niveau des ontologies que gère l'agent.
- Les tâches AjoutRelation/SupressionRelation permettent l'ajout et la suppression de relation entre deux concepts au niveau des ontologies que gère l'agent.
- Les tâches AjoutDataSource/SupressionDataSource permettent l'ajout d'une correspondance aux propriétés existantes au niveau des ontologies que gère l'agent.

Les protocoles relatifs à la gestion d'ontologies sont traités de manière plus détaillée dans le chapitre suivant.

VII.4.2 *Évaluation d'impacts*

L'**évaluation d'impacts** consiste à *évaluer l'importance qu'ont les changements au sein des sources de données sur les ontologies*. Comme nous l'avons expliqué dans le chapitre IV, section 2, l'impact des changements est une invalidation des correspondances définies entre la l'ontologie et les sources de données. La Figure 33 illustre un concept "Ingénieur équipement" composé de plusieurs propriétés. Les sources de données associées aux propriétés sont décrites entre des crochets ([...]). Les données relatives à la propriété "Nom" peuvent être récupérées dans la source de données "RH" et "ProdDb". Supposons que ces sources de données ont des sources de données relationnelles, dans ce cas "employé" et "personnelProd" représentent des tables. "name" et "nom_ing" sont des attributs de type chaîne de caractères.

Si, pour une raison quelconque, la table "employé" est supprimée, les données pourront toujours être récupérées dans la source "ProdDB".

Supposons maintenant que "ProdDB" n'existait pas, le fait de supprimer la table "employé" impliquerait qu'aucune donnée relative au concept "Ingénieur équipement", sauf le salaire, ne pourra être récupérée. Cette incapacité à récupérer les données du concept "Ingénieur équipement" implique qu'il n'est plus possible de récupérer un ensemble de données permettant d'apporter une réponse complète aux requêtes utilisateurs. Admettons que l'utilisateur souhaite récupérer toutes les opérations de maintenance faites sur les équipements par les ingénieurs équipement. L'impossibilité de récupérer les données relatives au concept « ingénieur équipement » impliquerait une réponse partielle à la requête utilisateur, à savoir les opérations de maintenance faites sur les équipements.

Ingénieur_équipement{
Nom, String, [(name, String, **employe**, RH) OU
(nom_ing, String, **personnelProd**, ProdDb)];
Prénom, String, [(pnom, String, **employe**, RH) OU
(pnom_ing, String, **personnelProd**, ProdDb)];
Address, String, [(no,String, **employe**, RH),
(road, String, **employe**, RH),
(zip_code, String, **employe**, RH),
(city,String, **employe**, RH) OU (address, String, **personnelProd**, ProdDB)];
Salaire, Double, [(salaireNet, double, **paie**, RH)]}

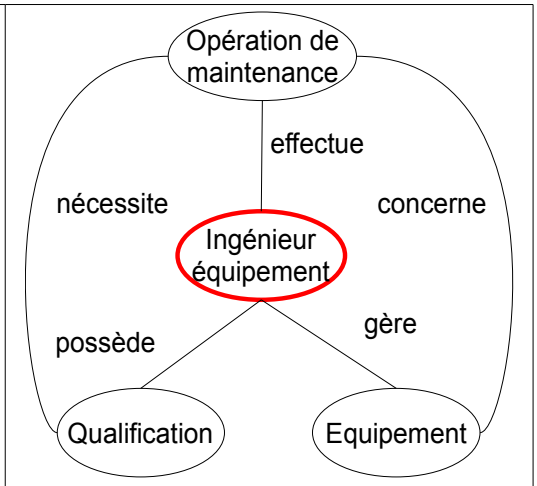

Figure 33: Impact des changements.

L'*agent ontologies* a pour tâche de traiter les notifications de changements envoyées par les *agents ressources*. Suite à cette **évaluation d'impact** (evaluerImpact), l'agent peut soit **mettre à jour les ontologies** (miseAjourCorrespondances) afin qu'elles soient cohérentes avec les changements effectués, soit **informer l'administrateur** (informerAdministrateur) qu'une ontologie est incohérente par rapport à une ou plusieurs sources de données. La *mise à jour de l'ontologie à lieu dans le cas de changements simples comme la suppression complète ou partielle d'une source de données. Dans le cas de changements plus complexes comme la fusion de structure une alerte est émise vers l'administrateur*. Nous verrons les différents protocoles mis en œuvre dans le chapitre suivant.

VII.4.3 *Conversion de requêtes*

En plus de ces tâches de gestion des ontologies, l'*agent ontologies* est aussi responsable de la reformulation de requêtes. Comme nous l'avons décrit dans la section 4 du chapitre VI, afin de pouvoir récupérer les données relatives aux connaissances modélisées, il est nécessaire de convertir les requêtes formulées sur une ontologie métier en requêtes compréhensibles par les sources de données (ConversionDeRequete). *C'est l'agent ontologies qui est responsable de la mise en œuvre des algorithmes décrits dans le chapitre VI.4.1. Il est aussi responsable de la reformulation de requêtes (reformulerRequete) en utilisant des sources secondaires si elles ont été définies (cf chapitre IV.3). Cette reformulation est effectuée, soit si la précédente reformulation a été invalidée par un des agents ressources, soit si l'une des principales sources de données est indisponible*. Suite à la conversion ou la reformulation, l'*agent ontologies* fait une demande de récupération de données (demandeDeDonées) afin que les agents ressources récupèrent les données des sources.

VII.4.4 *Navigation d'ontologies*

Finalement, c'est l'*agent ontologies* qui fournit aux agents utilisateurs le modèle de l'ontologie que l'utilisateur souhaite utiliser. *Ce modèle est une extraction des éléments de l'ontologie (extractionModèleOntologie) sous forme d'un document XML*. Ce modèle est envoyé à l'agent utilisateur (envoiDuModel). A réception du document, l'agent peut le traiter et afficher le contenu de l'ontologie. A partir de ce modèle, l'utilisateur métier peut naviguer au sein des concepts, propriétés et relations de l'ontologie.

VII.5 Veille des sources et récupération des données

Le module de veille des sources et de récupération des données est composé d'un ensemble d'*agents ressources, un pour chaque source de données*. Ces agents sont placés au niveau de chaque source de données pour faciliter la détection des changements. Les comportements des agents ressources sont décrits dans la Figure 34.

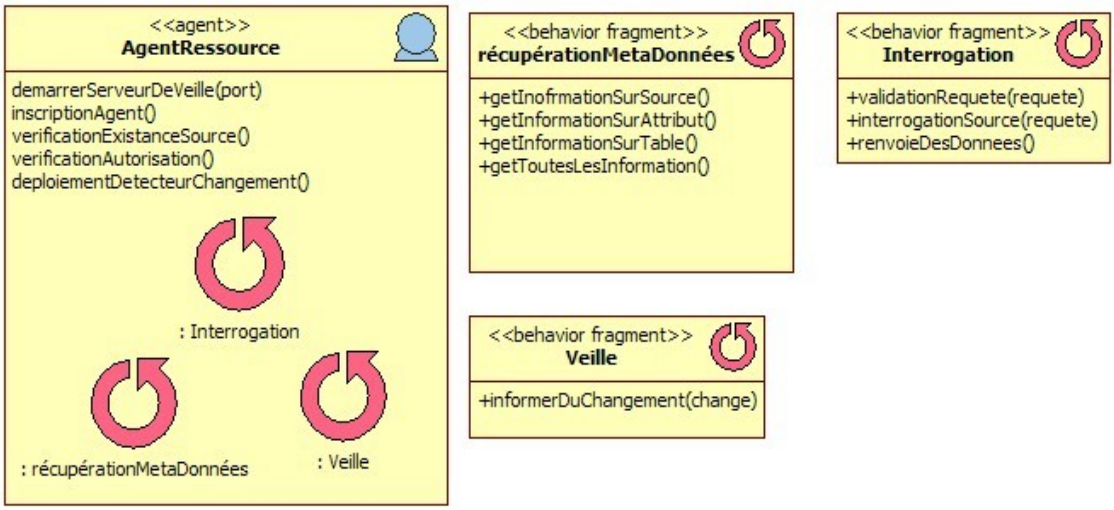

Figure 34: Architecture interne des agents ressource.

Les agents ressources ont trois principaux comportements :

- exécuter les requêtes d'interrogation de données
- renvoyer des métadonnées[10] sur les sources de données dont ils s'occupent
- surveiller les changements effectués au sein de leur source de données respectives.

VII.5.1 *Interrogation*

Les **agents ressources** sont responsables de l'*interrogation* des sources de données. Préalablement à cette tâche d'exécution de requête, ils valident les requêtes (validationRequete) adressées à leur source de données. En cas de requêtes invalides, une *demande de reformulation* est effectuée. Dans le cas contraire, l'agent *exécute et renvoie les données* (interrogationSource/renvoiDesDonnées).

VII.5.2 *Collecte des métadonnées*

Les **agents ressources** sont aussi responsables de la *collecte des métadonnées* relatives aux attributs qui sont ajoutés à la représentation des connaissances. *Ces métadonnées sont*

10 Les métadonnées correspondent à un ensemble d'informations techniques et descriptives. Par exemple les métadonnées sur un attribut présent dans une base de données relationnelle peuvent être son type, sa taille, la table à laquelle il appartient, est ce que c'est une clé primaire ...

utilisées lors de la création d'une ontologie ou lors de l'ajout de nouveaux concepts ou la redéfinition de certaines correspondances. Ils peuvent, par conséquent, récupérer des métadonnées sur la source de données (getInformationSurSource), sur les tables (getInformationSurTable), sur les attributs (getInformationSurAttribut) ou récupérer toutes les métadonnées (getToutesLesInformation).

VII.5.3 *Veille de la source de données*

En plus de cette tâche de collecte de métadonnées, les agents ressources sont aussi responsables de la veille des sources. *Le mécanisme de veille prend essentiellement deux formes*. La première correspond au **déclenchement d'une alerte à chaque modification** de structure de la source de données. La deuxième, consiste à **émettre une alerte si les éléments d'une requête reformulée ne sont plus présents** au sein de la source. Les alertes sont envoyées (informerDuChangement) à l'*agent ontologies* pour l'évaluation de l'impact.

VII.6 Services aux agents.

Le module « services aux agents » regroupe les agents de type annuaire, mais aussi un agent de veille. Les agents d'annuaire sont au nombre de deux et ils gèrent l'annuaire des agents et des sources de données présentes dans le système. Les services aux agents peuvent être implémentés en utilisant une autre technologie que les agents telle que les web services, CORBA. Mais *pour des raisons de simplicité et d'interopérabilité entre ces différentes technologies, nous avons choisi de les implémenter sous forme d'agents.*

VII.6.1 *Agent annuaire système*

L'**annuaire des agents** sert à *garder trace de tous les agents présents à un moment donné* et permet de faciliter la communication entre eux. En effet, il est possible à tout moment, pour des besoins de communication, d'obtenir l'adresse d'un agent donné en émettant une requête auprès de l'annuaire des agents. Ce dernier enregistre aussi les services qu'offrent les

différents agents. Par service, nous entendons les objectifs que les agents peuvent réaliser.

VII.6.2 *Agent annuaire sources de données*

L'**annuaire des sources de données** est quelque peu différent. Non seulement il *répertorie les sources de données* présentes dans le système, mais il permet aussi d'obtenir d'autres informations. Ces informations peuvent concerner l'état de la source de donnée, *est-ce qu'elle est en ligne ou hors-ligne, sa localisation, un bref descriptif de son contenu, l'administrateur de la source de donnée*. Ces informations sont renvoyées par les agents ressources et certaines informations sont régulièrement mises à jour. En plus de fournir les informations relatives aux sources de données, l'agent annuaire sources de données est aussi responsable de la coordination des agents ressources lors de la récupération des données que nous détaillons dans le chapitre suivant.

VII.6.3 *Agent veille*

L'**agent de veille** a pour tâche de *détecter d'éventuelles déconnexions des agents du système*. Cet agent *s'assure que tous les agents du système sont présents. D*ans le cas contraire, une alerte est émise auprès de l'administrateur système afin que des **actions correctives** soient prises. Les actions correctives peuvent être le *re-déploiement d'un agent* ressource suite à un *crash* ou la *vérification de la connexion réseau*.

VII.7 Module d'administration

Le module d'administration est composé d'un agent qui permet de prendre en compte deux types de modification : 1) les évolutions d'ontologie et 2) l'ajout ou la suppression de sources de données du SID.

VII.7.1 *Évolution d'ontologie.*

L'**évolution de(s) ontologie(s)** se fait généralement par *l'ajout ou la suppression de concepts ou de propriétés*. Comme nous l'avons vu en introduction de ce manuscrit, les connaissances modélisées sont susceptibles d'évoluer avec le temps. Lors de ces évolutions, les concepts peuvent devenir inutiles, de nouveaux concepts peuvent apparaître et d'autres peuvent être regroupés afin de former de nouveaux concepts plus pertinents pour la description du métier. Ces différentes opérations nécessitent des mises à jour de la représentation des connaissances. L*'ajout de nouveaux concepts n'a pas d'impact sur les correspondances existantes puisque de nouvelles correspondances sont créées. La suppression de propriétés ou de concepts implique une suppression des correspondances associées et le regroupement de concepts réutilise les correspondances déjà existantes.*

L'ensemble des tâches relatives à l'évolution d'ontologie(s) est déclenché par l'administrateur par l'intermédiaire de l'agent d'administration. La réalisation de ces évolutions nécessite la coopération du module gestion et manipulation des ontologies et du module veille des sources et de récupération de données. Le module gestion de manipulation des ontologies possède un comportement « MaintenanceDontologie » (cf. Figure 32) permettant la modification de la structure des ontologies. Le module veille des sources de données dispose, lui, d'un comportement « RécupérationMetaDonnées » (cf. Figure 34) qui permet le renvoi de métadonnées sur les sources d'informations

VII.7.2 *Évolution de sources de données.*

Dans le cas d'**ajout ou de suppression de sources de données** par l'administrateur du SID, il sera nécessaire de *créer ou de supprimer des correspondances*. *Ces nouvelles correspondances peuvent concerner des concepts existants, c'est-à-dire que la source de données apporte une information sur les concepts déjà existants, ou nécessiter la création de nouveaux concepts. La suppression de sources de données n'engendre pas de modifications importantes sur la représentation des connaissances.* Les correspondances faites à partir de la source de données supprimée sont elles aussi supprimées. Cette suppression de source de données peut entraîner la perte, pour certaines propriétés, de leur unique source de données. Dans ce cas, ces propriétés sont signalées à l'administrateur qui pourra définir une source de données alternative ou supprimer ces dernières.

Après avoir défini les modules ainsi que les agents qui composent notre SID , nous allons voir dans la section suivante le fonctionnement générale du système proposé. Nous verrons par la suite, en détail, les principaux protocoles d'interaction.

VII.8 Fonctionnement général

La coopération entre les agents des différents modules nous permet de réaliser les objectifs suivants : **détection des changements** au niveau des sources de données, **récupération des données** relatives à une requête utilisateur, et **évolution des représentations métiers** et des sources de données. Nous verrons de manière détaillée dans les sections suivantes les processus mis en place pour la réalisation de ces objectifs. *Les trois principaux points d'entrée du système sont la détection de changements, l'agent utilisateur et l'agent administrateur.* Nous illustrons dans la Figure 36 l'enchaînement des différents comportements des agents. Le premier point d'entrée (1) est la **détection d'un changement** au sein d'une source de données. Le traitement du changement nécessite la coopération de l'*agent ontologies* pour la mise à jour de l'ontologie. Le protocole de traitement du changement est explicité dans le chapitre suivant. Le deuxième point (2) d'entrée est la **demande d'une ontologie par l'agent utilisateur**. Cette demande est faite pour permettre à l'utilisateur d'explorer les connaissances contenues dans une ou plusieurs sources de données. L'*agent ontologies* renvoie un document XML représentant l'ontologie.

Cette exploration peut donner lieu à la formulation d'une requête pour **récupérer les données** qui constituent un troisième point d'entrée (3). Cette requête est envoyée à l'*agent ontologies* pour la reformulation. Cette dernière effectuée les sous-requêtes sont envoyées à l'agent annuaire source pour la coordination de la récupération des données par les agents ressources. Le protocole de récupération de données est détaillé dans le chapitre suivant.

Le quatrième point d'entrée (4) concerne la **mise à jour de l'ontologie**. Ces mises à jour peuvent nécessiter des métadonnées concernant les sources de données du système. Dans ce cas, une requête de demande de métadonnées est envoyée aux agents ressources concernés.

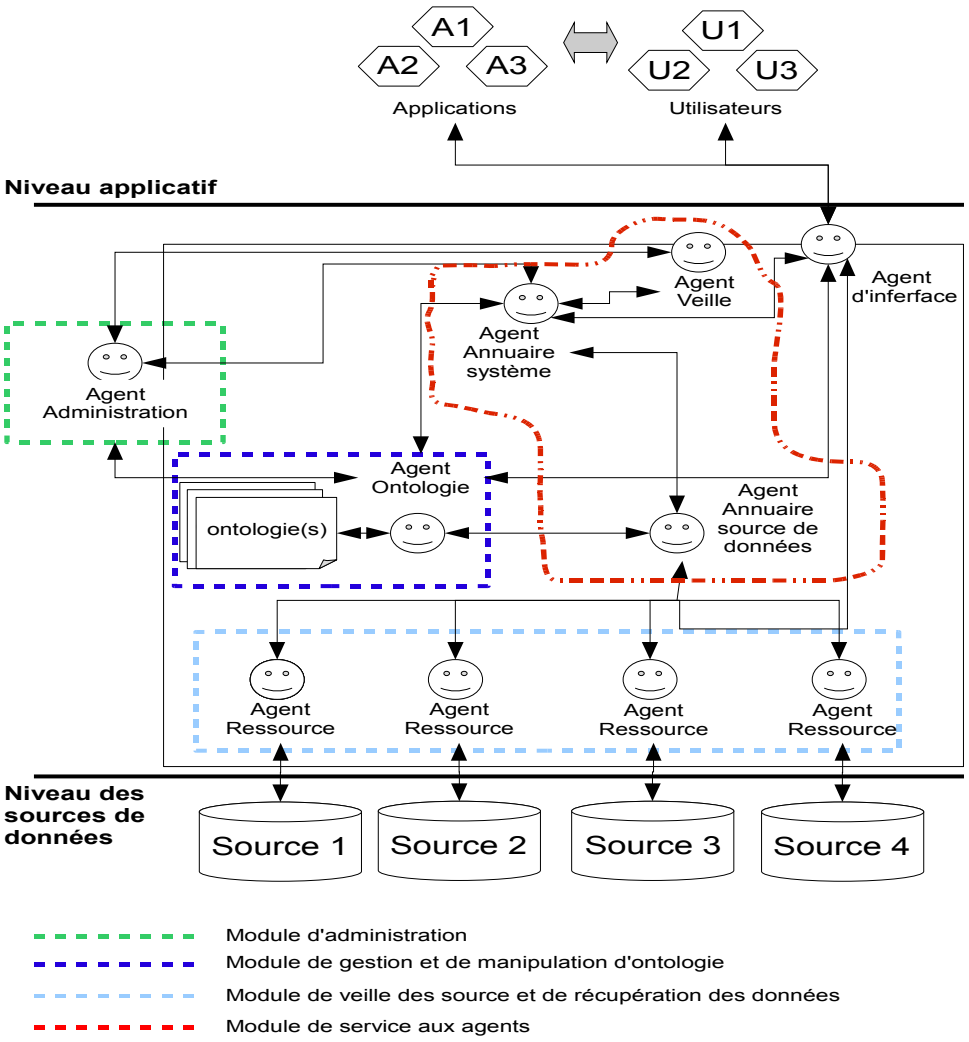

Figure 35: Fonctionnement général

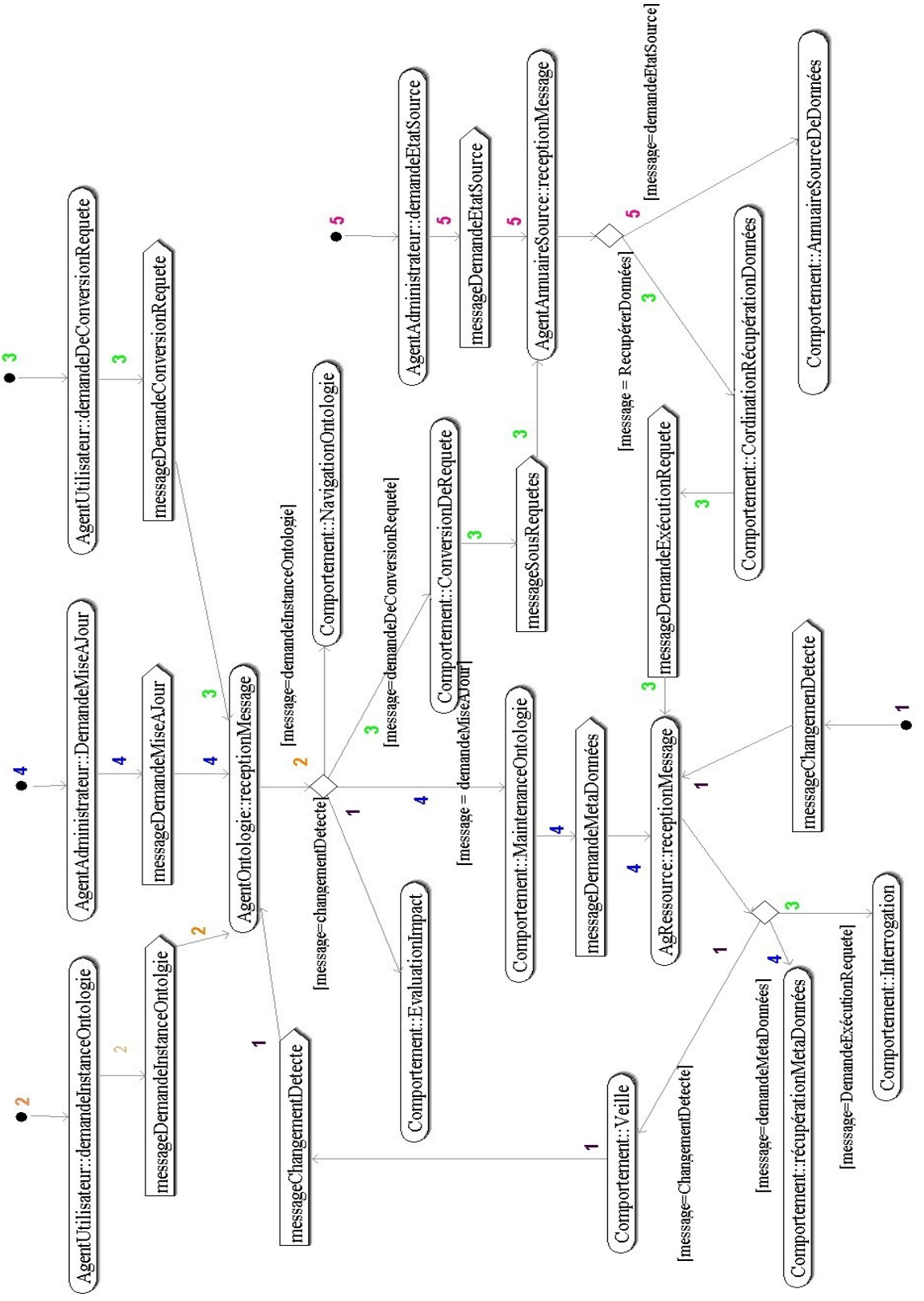

Figure 36: Enchaînement global des comportements

Le dernier point d'entrée (5) concerne la **demande de l'état des sources de données** par l'administrateur. Cette demande nécessite la coopération de l'agent annuaire source de données pour connaître l'état des sources de données participantes au SID. Le dernier point d'entrée concerne une demande de l'état des sources de données participantes au SID. Cette demande est effectuée par l'administrateur via son agent.

VII.9 Synthèse

Dans ce chapitre nous avons vu l'architecture générale de **KRISMAS** qui est basée sur des agents. L'architecture est composée de plusieurs modules qui permettent la *gestion des ontologies, la récupération de données et la veille des sources de données, l'administration du système et les services aux agents*. L'architecture que nous proposons permet l'exploitation des ontologies et apporte une réponse à la problématique de gestion du changement que nous avons mis en évidence dans la discussion du chapitre V. En effet, l'utilisation des agents nous permet de détecter les changements au niveau de chaque source de données. Cette détection nous permet, soit de mettre à jour l'ontologie, soit d'avertir l'administrateur des incohérences entre l'ontologie métier et les sources de données. Dans le chapitre suivant, nous nous concentrons sur l'adaptation du système à savoir les différents mécanismes que nous avons mis en place pour la détection de changements, la récupération de données et l'évolution des sources de données ou des représentations de connaissances.

Chapitre VIII.

Adaptation du système

VIII.1 Introduction

Nous avons vu dans le chapitre précédent notre proposition en termes de SMA comme support à l'intégration de données. Nous avons également décrit le fonctionnement général des modules qui composent KRISMAS. Dans ce chapitre, nous nous concentrons sur les principales tâches que sont la détection de changements, la récupération des données, et l'ouverture du système.

VIII.2 Détection des changements

La détection des changements est le résultat des activités de veille réalisées par les agents ressources. C'est la principale tâche de KRISMAS. Elle nous permet de vérifier si les représentations métiers sont toujours cohérentes avec les sources de données suite à une modification d'une ou plusieurs sources de données. Par cohérence de la représentation des connaissances métiers, nous entendons que les données peuvent être récupérées soit à partir d'une source de données principale, soit à partir d'une source de données secondaire. Comme

nous l'avons déjà expliqué, les changements au sein de sources de données peuvent rendre les correspondances invalides mais aussi rompre les liens, définis par l'administrateur du SID, entre les différentes sources de données.

VIII.2.1 *Principe*

Comme nous l'avons vu dans le chapitre III, les changements peuvent être de deux types : les changements de contenu d'une source de données et les changements de structure d'une source de données. Nous avons vu dans le chapitre II.2.1 que le changement de contenu résulte d'une utilisation courante de la source de données et n'a pas d'effet sur le SID. Dans le cas de changements de structure[11], les impacts sont plus importants. Ils invalident des correspondances définies entre la représentation des connaissances et des sources de données. Ces modifications de structure sont réalisées par des commandes de manipulation de structure. Ces commandes[12] forment le langage de définition de données (LLD). *La détection de l'utilisation de ce type de commandes peut se faire suivant deux méthodes : soit on vérifie que le schéma est le même à intervalle régulier, soit des messages sont émis lors de l'utilisation des commandes de définition de données.* Le processus de détection de changements est illustré par un diagramme d'activité suivant la notation UML dans la Figure 37 ci-dessous.

Nous avons choisi la **deuxième solution** dans la mesure où cette solution est *moins gourmande en ressources*. En effet, aucun traitement n'est effectué tant qu'une modification de la structure n'a pas été détectée. *L'envoi de message vers un agent ressource se fait en utilisant des déclencheurs. Les déclencheurs appellent des fonctions qui récupèrent les informations nécessaires et les envoient sous forme de messages à l'agent ressource*. A chaque modification de type "ALTER"[13] ou "DROP[14]" les messages sont envoyés (Figure 37:changementDetecte). Nous ne nous intéressons pas à la commande "CREATE" dans la mesure où elle se traduit par un ajout à la représentation des connaissances, alors que les deux premiers à une modification de structures.

11 on considère qu'ils ont lieu lorsque le modèle ne convient plus à la réalité et qu'il est nécessaire de le faire évoluer.

12 *Create table, alter table ou drop table*

13 *Alter* est une requête permettant la modification d'une table en ajoutant, en supprimant ou en changeant les colonnes, leur définition ou les contraintes. Nous ne nous intéressons pour le moment qu'à la suppression de la colonne.

14 *Drop* est une requête permettant la suppression complète d'une table.

Une fois le message vers l'*agent ressource* émis, l'*agent ressource* informe l'*agent ontologies*. Le contenu du message envoyé correspond à un document XML contenant les modifications effectuées. A réception du message, l'*agent ontologies* répertorie les propriétés, ainsi que les concepts qui sont concernés par le changement. L'importance qu'ont les concepts dans la représentation métier est ensuite évaluée (Figure 37 : evaluationImpact). Deux cas de figure se présentent. Soit l'agent est incapable de mettre à jour la représentation des connaissances et informe donc l'administrateur du système (Figure 37 : avertirAdministrateur). Soit en plus d'évaluer l'impact qu'ont les changements, l'*agent ontologies* peut aussi mettre à jour les représentations métiers concernées (Figure 37 : miseAJourOntologie) et informer l'administrateur des modifications effectuées (Figure 37 : avertirAdministrateur).

Après avoir défini le principe général, nous verrons dans la section suivante comment sont effectuées l'évaluation d'impacts et la mise à jour des ontologies.

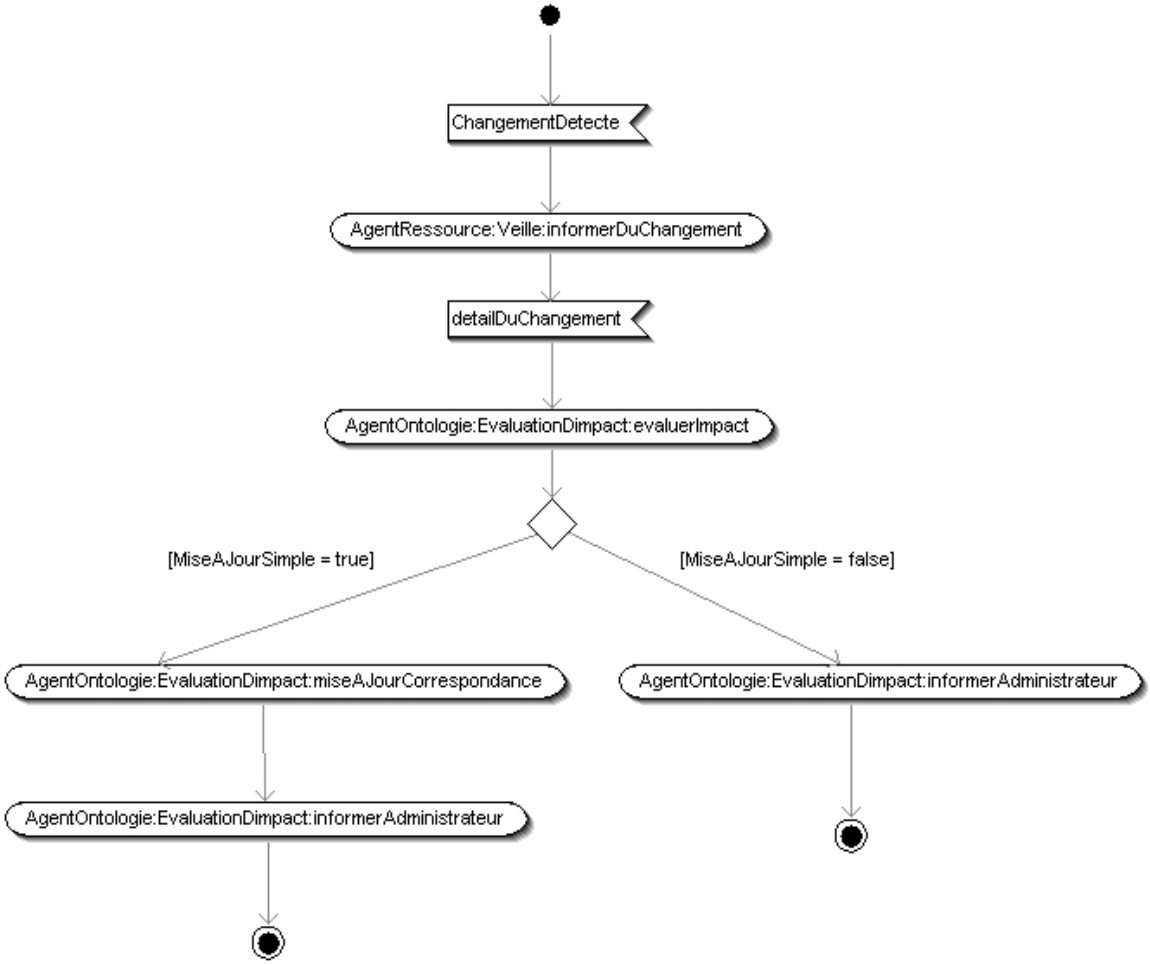

Figure 37: Détection de changements.

VIII.2.2 *Évaluation d'impacts et mise à jour*

Après avoir reçu une notification de changement au sein d'une source de données, l'*agent ressource* informe l'*agent ontologies* du changement. Ce dernier procède alors à une évaluation d'impact et à la mise à jour de la représentation métier.

L'évaluation de l'impact des changements est effectuée en trouvant les propriétés concernées. Elles sont trouvées à partir d'une matrice de correspondances. Cette matrice est construite automatiquement par l'*agent ontologie* à partir des correspondances qui ont été définies entre une représentation des connaissances et les sources de données associées. Les lignes de la matrice correspondent aux propriétés des différents concepts et les colonnes aux éléments des sources de données. Les relations entre propriétés et éléments des sources de données sont définies par la présence de 1 (0 pour l'absence de relations) à l'intersection des lignes et colonnes respectives. À partir de cette matrice, on peut trouver toutes les propriétés qui sont concernées par un changement au niveau d'une source de données.

La mise à jour des correspondances peut prendre deux formes. Dans le cas de suppression d'éléments d'une source de données, les définitions sont supprimées des propriétés concernées. Dans le cas où les changements s'apparentent à un renommage d'attribut, l'agent ontologies va identifier les correspondances qui sont influencées et remplacer l'ancienne définition par la nouvelle. Ces opérations sont réalisées par l'*agent ontologies* par la fonction de mise à jour des correspondances (miseAJourCorrespondance). Finalement, en cas de changements plus complexes comme la division ou la fusion d'attributs, une alerte est émise auprès de l'administrateur du système.

Une fois identifiés et mis à jour, ces concepts sont analysés pour déterminer s'il est toujours possible de récupérer les données relatives à ces derniers. En d'autres termes, nous recherchons s'il est possible, pour chaque propriété, de récupérer les données à partir d'une autre source de données. Si tel n'est pas le cas, nous évaluons l'importance qu'a le concept dans la représentation des connaissances. C'est à dire si le concept est un nœud central ou pas et combien de relations sont rompues. Par exemple, dans la Figure 38, s'il n'est pas possible de récupérer les données désignées par les attributs ξ associés aux propriétés du concept 2 les concepts 1,3,4, et 5 sont impactées du fait de leur relation avec le concept 2. Dans le cas du concept 8 l'impact est moindre dans la mesure où il ne possède qu'une relation. La Figure 39, illustre l'impact des changements de structure sur les relations inter-sources. Si les attributs $\xi^A_{1,}\xi^A_{2,}\xi^A_{8,}\xi^A_9$ sont supprimés de la source A, du fait de la suppression de leur structure, il sera impossible de définir des conditions de jointure pour la récupération de données.

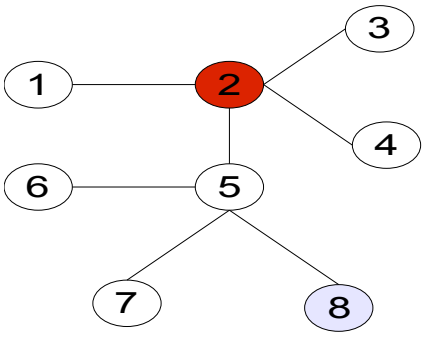

Figure 38: Impact sur les relations.

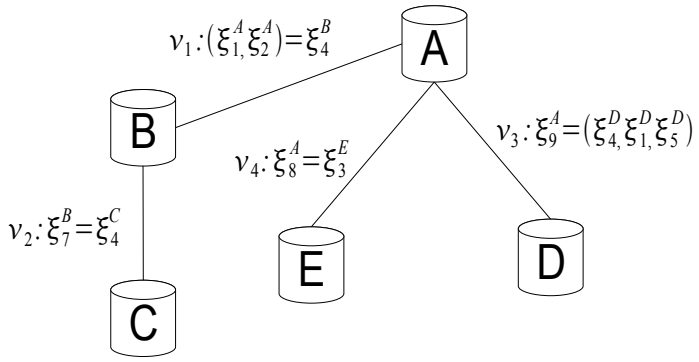

Figure 39: Impact sur les relations inter-sources.

Nous définissons, pour chaque catégorie d'impact, un indicateur qui prend la forme d'un

pourcentage. En effet, nous rapportons le nombre de concepts concernés par les changements au nombre total de concepts présents dans la représentation des connaissances. Le même calcul est fait pour les relations inter-sources. Ces deux indicateurs sont communiqués à l'administrateur.

Les changements ne sont pas les seuls événements susceptibles de rendre le système incohérent. En effet, des incohérences peuvent surgir lors de la récupération de données. En effet, si une requête formulée avant la mise à jour des correspondances, suite à un changement au niveau d'une source de données, est exécutée il est possible que les éléments demandés ne soient plus présents. L'exécution de ces requêtes peut être source de problèmes dans la mesure où les éléments de la requête peuvent ne plus exister. De plus, il est possible, dans un contexte d'environnement distribué, que la connexion entre plusieurs sites soit rompue. Afin de remédier à ces situations, il est nécessaire de prévoir un mécanisme de validation des requêtes et de reformulation.

VIII.3 Récupération des données

Comme nous l'avons déjà vu dans le chapitre V, la récupération se fait à partir de requêtes qui sont exprimées en fonction des éléments de la représentation des connaissances. À partir de ces requêtes nous avons proposé des algorithmes qui permettent de les reformuler en requêtes compréhensibles par les sources de données. *La reformulation, n'est pas une condition suffisante pour la récupération de données. D'une part, les sources de données contenant les données peuvent être indisponibles, d'autre part, tout changement au sein d'une des sources de données rend la reformulation invalide.* L'indisponibilité des sources de données peut être due soit à un retrait de la source de la données, soit à une indisponibilité temporaire liée à une mauvaise connexion réseau. Les requêtes obtenues, suite à la reformulation, peuvent contenir des structures ou attributs qui ne sont plus présents au sein des sources de données. Nous verrons dans cette section comment les agents sont utilisés pour remédier à ces situations

VIII.3.1 *Récupération de données*

Le processus de récupération de données est illustré par un diagramme d'activité décrit selon la spécification UML dans la Figure 40. Le processus commence par la formulation d'une requête, par l'utilisateur, à partir de la représentation des connaissances. Cette requête est communiquée au SMA, par l'intermédiaire de l'agent utilisateur qui agit comme point d'entrée au système. L'agent utilisateur recherche l'*agent ontologies* (trouverAgentOntologie) et lui demande de récupérer les données (demandeDeDonnées) correspondantes à la requête utilisateur. L'*agent ontologies* reformule la requête (reformulerRequete) utilisateur en un sous-ensemble de requêtes exécutables par les sources de données concernées. Cette reformulation se fait en utilisant les algorithmes définis dans le chapitre V. Les sous requêtes sont communiquées à l'agent annuaire des sources de données qui vérifie que les sources de données concernées sont bien en ligne (interrogationDesSources). Cette vérification est effectuée en envoyant un message aux agents ressources responsables des sources concernées. Si toutes les sources sont disponibles, les différentes requêtes sont adressées aux différents agents ressources (demandeExecutionRequete).

Le contenu des messages envoyés aux agents ressources prend la forme d'un objet avec les différentes parties de la requête. *Chaque agent ressource vérifie que les composants de la requête obtenue sont bien présents dans la source de données* (validationRequete). Par composant nous entendons, par exemple, pour une requête SQL, les éléments formant les clauses "SELECT", "FROM" et "WHERE". Si les requêtes sont valides, elles sont exécutées (ExecutionRequete). Les données obtenues sont mises en forme et renvoyées (RenvoieDesDonnées) à l'agent annuaire source de données qui est responsable de fusionner les différents ensembles avant de renvoyer le résultat final à l'agent interface.

Dans l'éventualité où une ou plusieurs sources de données ne sont pas disponibles ou qu'une ou plusieurs requêtes ne sont pas valides, un processus de reformulation est déclenché (reformulerRequete). Ce processus de reformulation consiste à réécrire la requête initiale en utilisant cette fois-ci les sources secondaires pour les propriétés dont les sources primaires ne sont pas disponibles.

La disponibilité des agents ressources est déterminée par l'agent annuaire source de données. Ce dernier envoie de manière régulière une demande d'information sur l'état de la source de données. Lors de ces demandes, l'agent annuaire source enregistre les délais de réponse de chaque *agent ressource*. À partir de ces délais, un temps de réponse moyen est calculé pour

chaque *agent ressource*. Lors des demandes d'exécution de requête, si un *agent ressource* n'a pas répondu dans un délai équivalent à deux fois le délai moyen alors la source de données est considérée comme inaccessible. Dans ce cas une demande de reformulation est adressée à l'*agent ontologies* et les requêtes auprès d'autres agents ressources sont annulées dans la mesure où le résultat du processus de reformulation peut identifier d'autres sources de données, et par conséquent les requêtes ne seront pas les mêmes.

Le même processus est déclenché en cas de requêtes invalides. À la réception des requêtes, chaque *agent ressource* vérifie la validité et informe l'agent annuaire source de données sur la validité des requêtes. Si au moins une des requêtes est invalide du fait de changements effectués pendant la création de la requête métier ou la reformulation, toutes les autres requêtes sont annulées et une reformulation est demandée à l'*agent ontologies*.

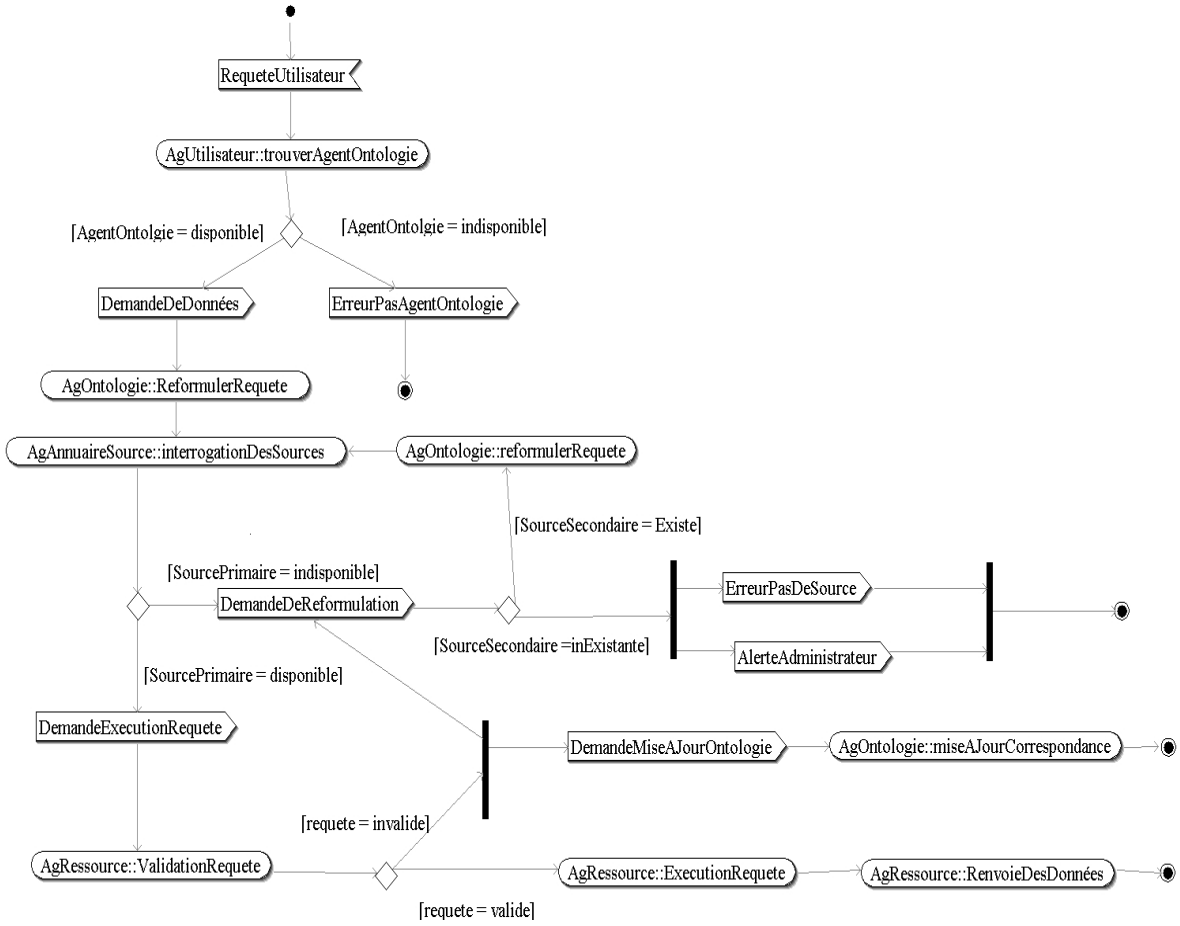

Figure 40: Récupération des données

VIII.3.2 *Reformulation de requêtes*

Comme nous l'avons précédemment décrit, plusieurs situations peuvent conduire à un besoin de reformulation d'une requête posée par un utilisateur à partir d'une représentation des connaissances. En effet, cette reformulation peut découler soit d'une invalidité d'une ou plusieurs sous-requêtes ou d'une indisponibilité d'une ou plusieurs sources de données.

La reformulation est effectuée par l'*agent ontologies* à partir des sources de données définies pour chaque propriété. Dans le chapitre V, nous avons expliqué que pour apporter une réponse aux requêtes Π utilisateur, les requêtes étaient décomposées en propriétés $p \in P$ et regroupées selon leur appartenance aux différentes sources de données. Chaque propriété est ensuite remplacée par sa définition $\psi \in \Psi$. Nous avons aussi défini, dans le chapitre précédent, qu'une propriété peut avoir une ou plusieurs correspondances $\theta \in \Theta$. *Pour les besoins de reformulation, nous utilisons les correspondances secondaires.* En effet, à la réception d'une demande de reformulation, les éléments invalides ou les sources de données indisponibles sont indiqués. Il est alors possible de déterminer les éléments de la requête initiale à remplacer. Une fois les éléments invalides identifiés, nous vérifions s'il existe une correspondance secondaire. En cas de présence de sources de données alternatives, la requête utilisateur est retraduite en utilisant les algorithmes définis dans le chapitre VI afin d'obtenir un nouveau sous-ensemble de requêtes. Cet ensemble sera alors adressé aux agents ressources pour la récupération des données. *Dans l'éventualité où il n'existe pas de sources de données alternatives, une alerte est émise auprès de l'administrateur et les propriétés invalides sont désactivées. Par désactivation nous entendons qu'un utilisateur ne pourra plus sélectionner une propriété qui ne dispose pas de source de données correspondante.* Il est aussi possible, en cas de source de données unique, d'informer l'utilisateur et de lui proposer une réponse partielle.

Si nous reprenons l'exemple de la Figure 33, après reformulation d'une requête qui récupère l'ensemble des données du concept « Ingénieur_equipement » une requête sera créée. Cette requête sera adressée à la source de données RH. Supposons que la table « employé » est supprimée ou renommée. L'*agent ressource*, lors de la validation, détectera que cette table n'existe pas dans la source de données et enverra un message d'invalidité à l'agent annuaire source de données. Ce dernier émettra une demande de reformulation à l'*agent ontologies*. Le résultat de la reformulation de l'*agent ontologies* sera cette fois-ci deux requêtes, une sur « prodDB » et une sur « RH » avec l'utilisation d'une règle pour relier les éléments des deux

sources de données.

Afin de remédier au problème d'absence de sources de données, l'administrateur système peut soit définir de nouvelles correspondances à partir de sources déjà participantes au SID ou décider de rajouter de nouvelles sources de données. Il peut aussi décider que certaines propriétés sont obsolètes et qu'une réorganisation de l'ontologie est nécessaire. Nous verrons dans la section suivante le principe ainsi que la méthode utilisée pour faire évoluer les représentations des connaissances, mais aussi comment rajouter ou supprimer des sources de données au SID.

VIII.4 Ouverture du système

Dans cette section, nous détaillons notre proposition sur l'évolution des connaissances modélisées, mais aussi l'évolution des sources de données. Nous rappelons que l'*évolution des connaissances prend la forme d'une modification au niveau des ontologies définies*. Ces modifications peuvent prendre la forme d'ajout (suppression), fusion (division) et le renommage de concepts, propriétés et de relations. Dans le cas des *sources de données, nous nous intéressons à la scalabilité du système c'est-à-dire à la facilité d'ajout et de suppression de sources de données*. En effet, la gestion du changement que nous proposons prend en compte les changements au sein des sources de données faisant partie du SID. L'ajout ou de suppression de sources de données est nécessaire si l'on souhaite enrichir le SID.

VIII.4.1 *Ajout/suppression de sources de données*

L'*ajout d'une source de données nécessite le déploiement d'un agent ressource*. Le processus de déploiement de ce dernier est illustré dans la Figure 41. L'administrateur commence par l'instanciation d'un *agent ressource* (instanciationAgentRessource). Ce dernier est déployé au niveau d'une source de données. Lors du déploiement, les paramètres relatifs à l'emplacement de la source de données ainsi que les paramètres de connexion sont chargés à partir d'un fichier de configuration au format XML. L'agent ressource vérifie l'existence de la source de données (VerificationExistanceSource) ainsi que les paramètres de connexion

(VerificationAutorisation). Ces vérifications effectuées, l'*agent ressource* peut déployer, au sein de la source de données, le mécanisme de détection des changements (deploiementDetecteurChangement).

L'ajout et la suppression de sources de données sont des actions effectuées de manière semi-automatiques par l'administrateur du système. L'ajout d'une source de données au SID peut se faire soit au déploiement de l'*agent ressource* ou à la demande de l'administrateur. Dans le premier cas, les métadonnées sont récupérées et renvoyées (recuperationMetaDonnées) en parallèle au déploiement du mécanisme de détection de changement. Dans le second cas, l'administrateur fait expressément une demande à un *agent ressource* afin de récupérer toute ou partie des métadonnées. Les informations envoyées à l'*agent ontologies* sont utilisées pour la création d'instances d'attribut (cf. chapitre VI section 2). L'administrateur peut alors associer aux propriétés déjà présentes les nouveaux attributs. Cette dernière opération est effectuée manuellement.

La suppression de source de données consiste à supprimer toutes les instances d'attribut appartenant à cette dernière. Lors de cette suppression, il est nécessaire de vérifier si la source de données n'est pas la source unique de données pour certaines propriétés. Dans l'éventualité où elle l'est, l'administrateur en est informé afin de soit définir d'autres sources de données, soit supprimer les propriétés concernées. La dernière tâche consiste à détruire l'*agent ressource*. Ce dernier mettra fin à ses activités dès que toutes les requêtes en attente auront été traitées.

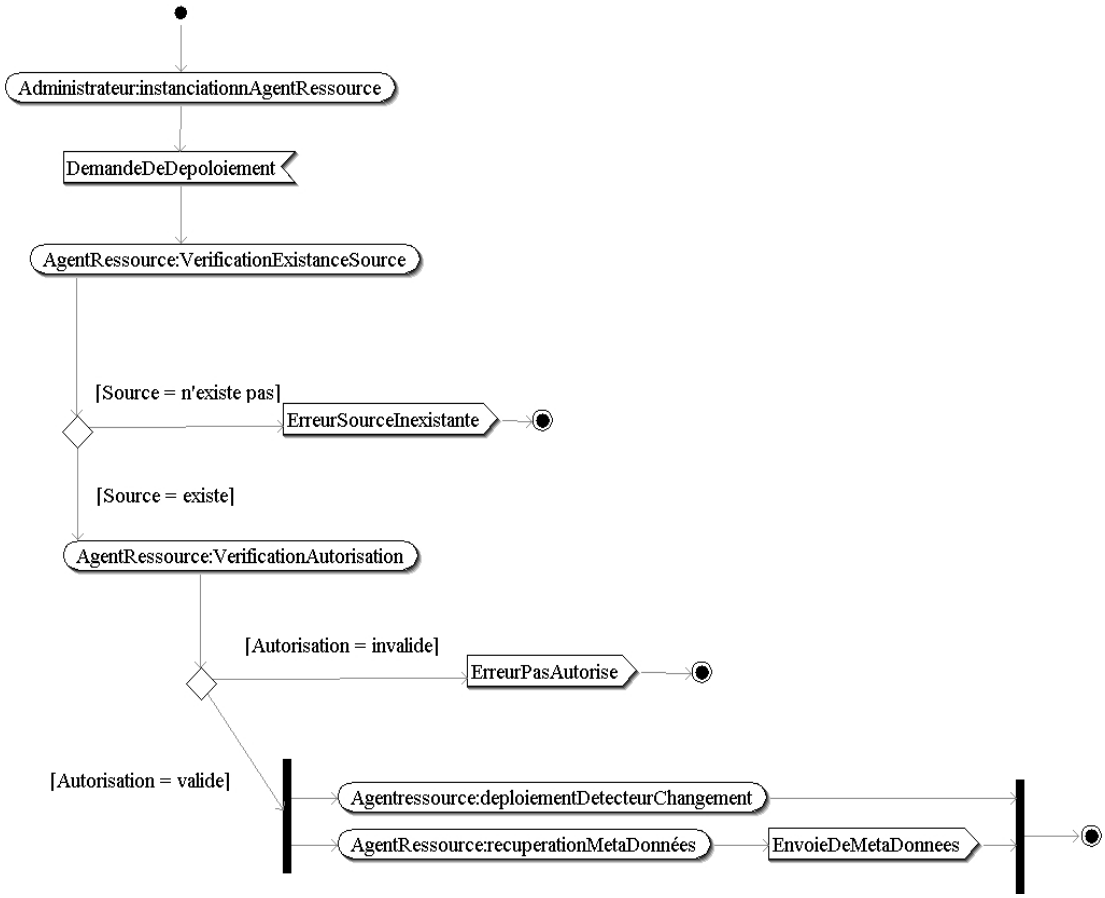

Figure 41: Déploiement d'un agent ressource

L'administrateur intervient donc dans le choix des métadonnées à importer lors de l'ajout d'une source de données, mais aussi dans l'association des correspondances à la représentation métier. Il intervient aussi lors de la suppression de sources de données afin de mettre à jour la représentation des connaissances par des opérations de mise à jour.

VIII.4.2 *Évolution des connaissances.*

Les évolutions peuvent être dues, soit aux évolutions au sein des sources de données, soit aux évolutions du domaine métier. *Quelque soit l'origine de l'évolution, les opérations effectuées sur la représentation des connaissances se font par l'intermédiaire de l'agent administrateur*. En effet, pour chaque modification, l'agent administrateur enverra un ordre de mise à jour à l'*agent ontologies*. Ce dernier exécutera l'ordre de mise à jour avant de communiquer à l'agent d'interface la (les) représentation(s) des connaissances actualisée(s).

Les **ajouts de concepts**, propriétés et de relations sont des opérations relativement simples à réaliser. Elles consistent en effet, à *créer une instance selon ce qu'on souhaite ajouter*. Afin d'être utilisable, l'administrateur du système doit impérativement définir, pour les propriétés composant les concepts, un ensemble de sources contenant les données. Ces correspondances avec les sources de données peuvent être déjà présentes dans l'ontologie ou créées par l'administrateur en ajoutant une nouvelle source au système.

Dans le cas de la **suppression des éléments** de l'ontologie, il sera *nécessaire de vérifier si les correspondances associées sont toujours utilisées*. Si elles ne le sont pas, le système propose à l'administrateur de les supprimer.

La **fusion/division** consiste à réorganiser les propriétés au sein des concepts. *Certaines propriétés peuvent être supprimées et les propriétés restantes ajoutées à un autre concept. Seule la suppression de propriétés a une incidence sur la représentation des connaissances dans la mesure où les correspondances associées ne seront peut être plus utilisées*. Il sera alors nécessaire de vérifier si les sources de données auxquelles appartiennent les attributs inutiles sont toujours utilisées.

Le **renommage** de concepts et de propriétés sont des opérations simples à réaliser et elles n'*ont pas d'influence sur les correspondances définies*.

L'ensemble de ces évolutions peuvent invalider les requêtes en cours de composition. Afin d'éviter ces invalidités, un processus de validation est réalisé avant la reformulation par l'*agent ontologies*. Ce processus consiste à valider la présence des éléments de la requête dans la représentation des connaissances avant la reformulation.

VIII.5 Synthèse

Dans ce chapitre, nous avons décrit l'architecture utilisée pour exploiter les représentations de connaissances définies dans le chapitre V. L'architecture que nous avons proposée est fondée sur un SMA et permet la détection de changements au niveau des sources de données ainsi que l'évaluation d'impact des changements sur la (les) représentation(s) de connaissances. Afin de récupérer les données, il est nécessaire de prendre en compte les éventuels changements au niveau des sources de données. Nous avons proposé, pour résoudre ce

problème, un protocole de reformulation qui permet en cas de requêtes invalides ou d'indisponibilité des sources de données de reformuler les requêtes utilisateurs. Cette reformulation se fait à partir des correspondances secondaires spécifiés au niveau de la représentation des connaissances. Finalement, nous avons proposé plusieurs protocoles pour faire évoluer le système, soit en termes de sources de données, soit en termes de représentation des connaissances.
Nous verrons, dans le chapitre suivant, la mise en œuvre de notre proposition sur un cas d'étude proposé par STMicroelectonics.

VIII.6 Conclusion

Nous avons vu, dans la partie précédente, les solutions existantes permettant de concevoir les différentes parties d'un SID. Selon nous, ces approches existantes présentent les inconvénients suivants :

- d'exposer le contenu des sources de données sans l'expliciter,
- de ne pas suffisamment prendre en compte les évolutions du domaine métier et des sources de données,
- de définir des correspondances peu flexibles entre le schéma global et les sources de données,
- de ne prévoir aucune source de données de secondaire en cas d'indisponibilité des sources de données.

Nous avons proposé dans cette partie, les différentes briques de KRISMAS permettant une modélisation de plus haut niveau des sources de données. Cette modélisation prend la forme d'une représentation métier du contenu des sources de données. Afin de permettre la récupération des données à partir de la représentation des connaissances, nous pouvons associer à chaque propriété un ensemble de correspondances. Ces correspondances représentent les différentes sources de données à partir desquelles il est possible de récupérer les données. Pour ce faire, nous avons proposé un algorithme qui permet de passer de la connaissance métier modélisée aux données. Nous avons aussi proposé une architecture à

base d'agents qui nous permet d'effectuer une reformulation de requête. Cette reformulation est effectuée, soit si une ou plusieurs sources de données n'est pas disponibles, soit si une partie des requêtes n'est pas valide. La reformulation utilise les sources secondaires définies au niveau des correspondances. Afin de garantir que les correspondances définies permettent de récupérer des données, nous avons proposé un mécanisme de détection de changements. Ce mécanisme permet aux agents ressources de déclencher une évaluation d'impact à chaque changement de structure effectués au niveau des sources de données. Dans le cas de changements simples la représentation des connaissances est mise à jour par l'*agent ontologies,* dans le cas contraire la représentation des connaissances est mise à jour par l'administrateur.

Chapitre IX.

Implémentation

Après avoir vu dans les trois chapitres précédents, nos propositions en termes de modélisation de connaissances ainsi qu'en termes d'architecture SMA, nous verrons dans ce chapitre comment nous avons appliqué notre proposition à un cas pratique proposé par STMicroelectronics. Nous commençons par préciser le contexte industriel et applicatif, avant de préciser le processus suivi pour la mise en place de nos propositions

IX.1 Contexte

Dans cette section, nous précisons le contexte industriel de STMicroelectronics mais aussi le contexte applicatif. Par contexte applicatif nous entendons les applications qui composent notre cas pratique ainsi que leurs rôles au sein du système d'information de STMicroelectronics.

IX.2 Contexte industriel

La société STMicroelectronics est une société multinationale franco-italienne, présente en

Europe, en Afrique, en Amérique et en Asie, spécialisée dans le domaine de la microélectronique. Elle est l'un des leaders mondiaux dans le développement et la fabrication de solutions semi-conducteurs. Elle offre plus de 3000 types de produits à plus de 1500 clients grâce à ces 50 000 employés.

STMicroelectronics est implantée sur 16 sites de fabrication, et notamment sur le site de Rousset (Bouches du Rhône, France) et possède sur ce site deux unités, une unité de production en 8" (correspondant à la dimension des plaquettes de silicium utilisées pour la production) et une unité de tests sur silicium.

Le système d'information de STM s'est développé par « morceaux » avec une approche parcellaire, par métier, qui n'a pas été dès sa conception intégré dans un schéma d'ensemble. Cette évolution, avec une approche locale, a donné lieu à un parc applicatif très complexe au regard du nombre d'applications gérées (une soixantaine uniquement rapportées au manufacturing) avec, comme conséquence, une multiplication des interfaces entre ces applications, mais aussi une multiplication des sources de données.

Nous nous concentrons, comme indiqué en chapitre 1, sur les aspects données. **L'objectif est de mettre en place un point d'accès unique aux sources de données de référence**. Ce point d'accès unique a pour objectif de garantir, au niveau applicatif, la validité des données utilisées. Au niveau des données, l'objectif est de réduire les mises à jour de données, tout en ayant une meilleure traçabilité des évolutions au niveau des sources de données. Au niveau des utilisateurs, l'objectif est d'avoir une meilleure visibilité quant au contenu des différentes sources de données.

Ce projet a été lancé par l'équipe « Architecture et Industrialisation » qui fait partie de l'organisation centrale chargée des orientations stratégiques du système d'information lié à la fabrication des plaquettes de silicium. Face à la complexité du système d'information et à la nécessité de mieux répondre aux besoins de leurs clients, les acteurs du système d'information ont ressenti la nécessité de mieux gérer les évolutions introduites dans le système.

IX.3 Contexte applicatif général

Nous nous situons au niveau du système de gestion de production (MES), et plus

particulièrement, dans le domaine de la planification d'entretiens des équipements de production. Les différentes applications responsables de cette planification sont décrites dans les sections suivantes.

IX.3.1 *Tracer Track.*

L'application *Tracer Track permet de gérer la certification des opérateurs et des techniciens sur les équipements et ou procédés*. En d'autres termes, il s'agit du système qui permet de répondre aux questions relatives à la qualification d'une personne donnée. Le système nous permet de savoir si un opérateur ou technicien donné est qualifié (ou a reçu la formation nécessaire) pour utiliser un équipement ou un procédé précis, et, les qualifications étant généralement délivrées pour une certaine période de temps, si la qualification est valide. La Figure 42 illustre les inter-connexions de Tracer Track avec les autres applications du système d'information. Les données qui peuplent Tracer Track sont issues du système de gestion des ressources humaines (PeopleFirst) et du système de gestion des congés (Arhistote). En plus des données du personnel de production, des données de production sont aussi ajoutées. Ces données de production proviennent de la base de centralisation du système de gestion de production (LDM). En résumé les principaux services offerts par Tracer Track sont de connaître:

- la qualification d'une personne sur tel équipement ;
- la qualification d'une personne sur tel poste de travail.

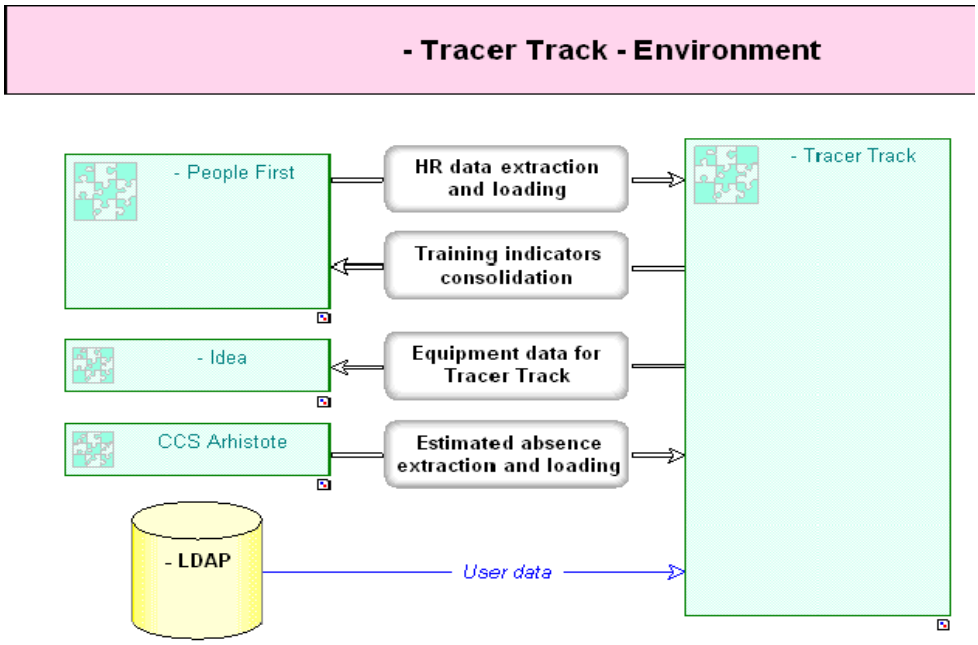

Figure 42: Tracer Track

IX.3.2 *TPMCenter*

L'application *TPMCenter (Total Preventive Maintenance Management Center) permet la gestion des tâches de planification de la maintenance des équipements de fabrication*. Comme l'illustre la Figure 43, cette application produit les plannings de maintenance préventive en utilisant les données portant sur les statuts de qualifications (issues de Tracer Track), l'état des équipements (application CAM), les compteurs équipements (nombre de plaques wafers fabriqués, …), informations de planifications (STAP) et les informations ADCS66 (Gestion de la documentation) qui permettent de renseigner sur les opérations à faire durant les tâches de maintenance.

Parmi les principaux services rendus par cette application, nous pouvons citer :

- la gestion des listes des opérations de maintenance effectuées (Checklists Management) ;
- la planification à base de compteurs (Counter Based Planning) ;
- la gestion de la maintenance curative (Curative Maintenance Management) ;
- l'extraction de compteurs équipements (Equipments Counter Files Extraction) ;

- le chargement des compteurs équipements (Equipments Counter Files Loading) ;
- la planification basée sur la fréquence (Frequency Based Planning) ;
- la gestion de la maintenance non-standard (Non Standard Maintenance Management) ;
- la gestion de la maintenance préventive (Preventive Maintenance Management) ;
- et la génération de fichiers STAP67 (STAP File Generation).

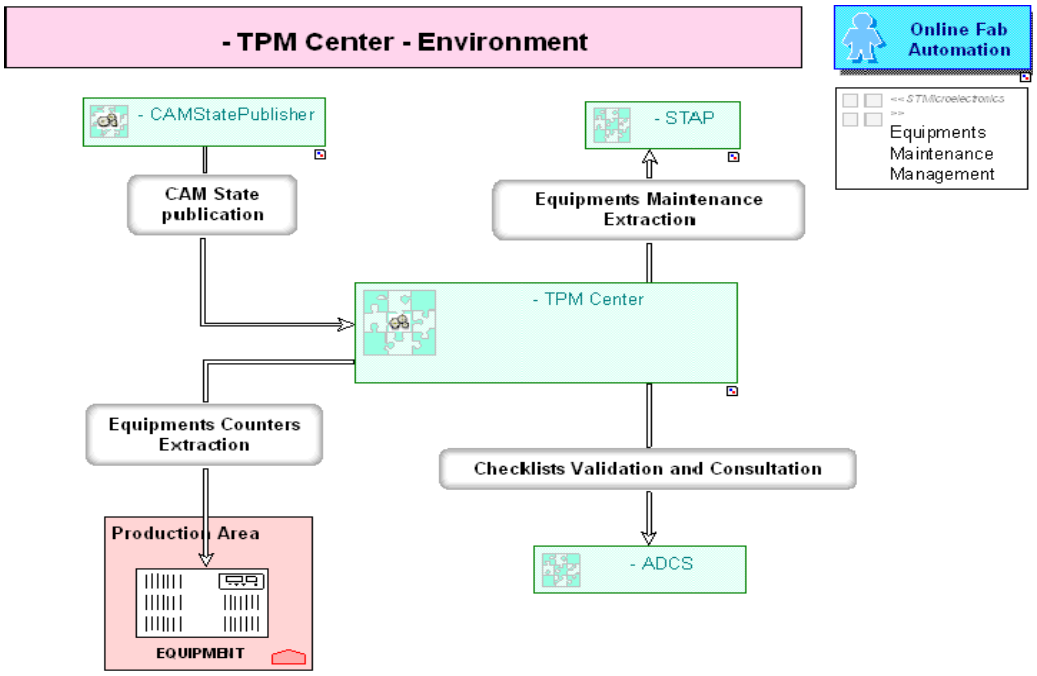

Figure 43: TPM Center

IX.3.3 *Problématique du cas test*

Le problème qui se pose dans ce cas pratique est que *les données contenues dans TPMCenter relatives aux personnels de production, ainsi que les données relatives aux équipements, sont saisies manuellement alors qu'elles existent dans des sources de données de références.* Plusieurs problèmes se posent.

Le plus évident est celui de la **validité des données**. En effet, le fait de saisir les données manuellement introduit le risque d'erreur de saisie. Une fois ces données saisies, il est nécessaire de **les maintenir à jour**. Cette mise à jour peut être fastidieuse dans la mesure où il sera nécessaire de **répertorier les changements** effectués au niveau de Tracer Track et

d'identifier les données impactées au sein de TPMCenter.

Un deuxième problème concerne la correspondance entre les données. En effet, *il n'existe pas de règles standard pour identifier une même chose au niveau des différentes applications*. Cette hétérogénéité au niveau de l'identification des données complique les opérations de mise à jour des données.

En plus de ces deux principaux problèmes, **plusieurs améliorations sont souhaitées**. La première consiste à mettre à la disposition des utilisateurs *un outil leur permettant de récupérer toute ou partie du contenu des sources de données de référence* pour des besoins d'analyse. La deuxième consiste à définir des *services permettant à un chef d'équipes de connaître la disponibilité des membres de son équipe pour d'éventuelles tâches de maintenance*. Cette disponibilité est définie en fonction de la présence physique, mais elle est aussi définie par rapport aux qualifications que ces personnes possèdent. Finalement, le dernier service concerne *la planification des formations afin de requalifier le personnel dont les qualifications expirent*.

Nous allons voir dans les sections suivantes le processus suivi pour la modélisation des connaissances et la mise en place du SMA

IX.4 Prototypage

Nous verrons dans cette section, les détails techniques liés à l'implémentation de KRISMAS en termes de modélisation de connaissances ainsi qu'en termes d'implémentation d'architecture.

IX.4.1 *Implémentations du modèle de connaissances*

Nous avons défini dans le chapitre V, les différents composants de notre modélisation. Nous le rappelons, elle est composée de concepts, de propriétés, de relations et de contraintes. À chaque propriété est associé un ensemble de sources de données. *Nous avons modélisé le domaine métier associé à l'application TMPCenter et TracerTrack sous forme d'une*

ontologie en OWL-DL. Cette ontologie a été *construite avec l'aide de Protégé*[15], un outil de construction d'ontologies.

Afin de modéliser le domaine métier, nous avons procédé à plusieurs *entretiens* avec les responsables de ces applications. Lors de ces entretiens, nous leur avons demandé de nous décrire leurs activités. À partir de ces différents entretiens, nous avons *identifié les différents concepts ainsi que les relations qui existaient entre eux. Ces concepts et relations ont ensuite été affinés et validés comme étant représentatifs du domaine métier*. Une fois les concepts obtenus, nous avons ensuite demandé quelles étaient les propriétés pertinentes pour le métier en question. La réponse à cette question nous a permis de détailler les différents concepts ainsi que les contraintes identifiées auparavant.

Ces informations nous ont permis de construire notre ontologie en OWL-DL à partir de Protégé. L'ontologie est illustrée dans la Figure 44 suivante.

15 http://protege.stanford.edu/

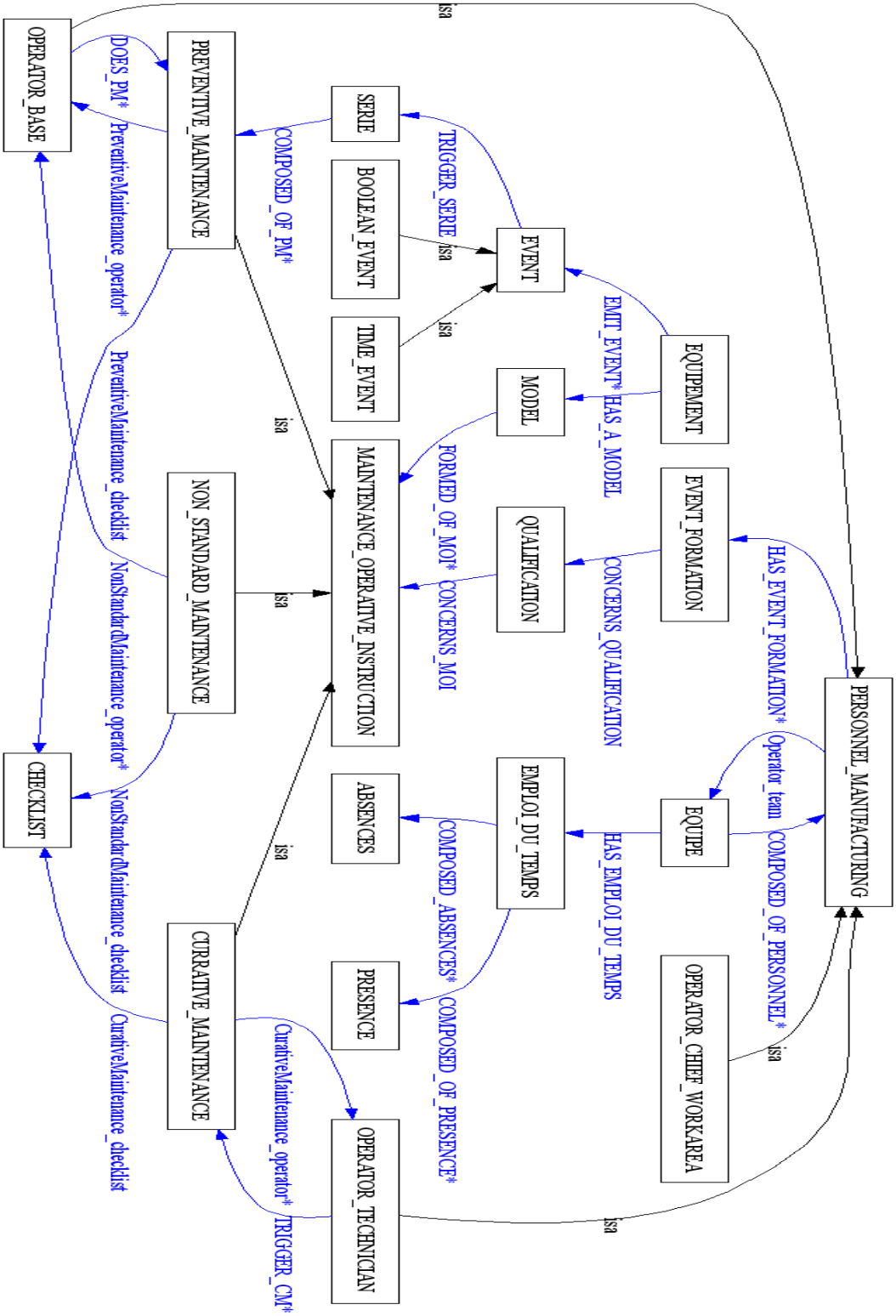

Figure 44: Modélisation des connaissances métiers.

Le domaine métier modélisé, nous nous sommes attachés à *identifier les sources de données de référence pertinentes*. Par pertinentes, nous entendons des sources de données de référence contenant des données relatives à une ou plusieurs propriétés ou concepts. Ces sources de données identifiées, nous avons *importé les métadonnées* relatives aux sources de données (url d'accès, type de la source, nom) ainsi qu'aux attributs (nom de l'attribut, son type, la structure d'appartenance). Ces métadonnées servent à *instancier une métaclasse* que nous avons définie dans notre ontologie. Ces instances formeront ce que nous avons défini dans le chapitre VI en tant que correspondances. La *métaclasse étend le rdf:property* ce qui implique que l'ontologie définie devient une ontologie en OWL-FULL. Ce changement n'a pas d'impact sur la vérification de la consistance de la représentation des connaissances dans la mesure où il est toujours possible de la rajouter avant les correspondances.
La dernière tâche est *d'associer à chaque propriété un ensemble de correspondances*. Comme nous l'avons déjà indiqué, cette opération est réalisée manuellement.

IX.5 Implémentation de l'architecture agent

Dans les chapitres VI et VII, nous avons détaillé notre proposition en termes d'architecture agent. Nous verrons dans cette section, comment nous avons implémenté nos propositions. Nous verrons, dans les sections suivantes, les outils utilisés et quelques spécifications techniques.

IX.5.1 *API*

Notre prototype est composé de plusieurs API qui nous permettent de manipuler (modifier, interroger) les ontologies définies, implémenter et déployer des agents, générer et manipuler des fichiers XML.

IX.5.1.1 Jena

Jena[91] est une API Java permettant la construction d'applications liées au Web Sémantique.

Elle permet d'utiliser les standards W3C tels que RDF, RDFS, OWL et SPARQL. L'api Jena est utilisée au niveau de l'*agent ontologies*. *Elle permet d'implémenter les modifications telles que l'ajout, la suppression ou la modification de concepts, propriétés ou relations*. Jena nous permet aussi d'effectuer des requêtes SPARQL sur l'ontologie. Ces requêtes permettent à l'*agent ontologies* de récupérer les informations nécessaires pour la reformulation de requêtes utilisateur. Nous pouvons en effet récupérer les informations relatives aux correspondances (nom de la source de données, nom des attributs, type) et les utiliser pour la mise en œuvre des algorithmes proposés.

IX.5.1.2 Java Agent Developpement framework (JADE)

JADE est une API Java qui facilite *le développement et le déploiement d'agents par l'intermédiaire d'un intergiciel conforme aux spécifications de la FIPA*[16][92]. JADE offre plusieurs services :

- une console d'administration afin de gérer les agents qui peuvent être répartis sur plusieurs machines ;
- un mécanisme de persistance des agents ainsi que des messages échangés ;
- un mécanisme de tolérance aux pannes.

Tous les agents de notre système sont implémentés sous forme d'agents Jade. Les messages qui sont échangés entre les agents sont des messages ACL[17] qui peuvent contenir des chaînes de caractères ou des objets.

IX.5.1.3 JDOM et Qizx

JDOM et Qizx[18] sont deux APIs qui permettent pour, l*a première, de générer et manipuler des fichiers XML, et, pour la seconde, d'effectuer des requêtes XQUERY* sur un document XML. Les fichiers XML sont utilisés au niveau des agents ressources. *XML est utilisé pour la structuration des fichiers de paramètres de lancement nécessaires à chaque agent ressource*. XML est aussi utilisé pour la *sérialisation des résultats obtenus lors de l'interrogation des sources de données*. Le résultat obtenu lors de l'interrogation d'une source est sous forme

16 Foundation for Intelligent Physical Agents
17 Agent Communication Language
18 http://www.xfra.net/qizxopen/

d'objet (ResultSet). Cet objet contient non seulement les données, mais aussi des métadonnées associées. Afin de ne garder que ce dont nous avons besoin, nous extrayons de l'objet les informations nécessaires et nous les enregistrons sous forme d'un document XML qui est communiqué aux autres agents. Comme nous l'avons précisé dans le chapitre VI, afin d'apporter une réponse aux requêtes utilisateur, il est tout à fait possible d'obtenir un ensemble de requêtes. Ces différentes requêtes sont, après interrogation de la source de données, stockées sous forme de documents XML. *Afin de présenter un résultat unifié aux utilisateurs, nous utilisons XQUERY pour fusionner l'ensemble des résultats obtenus*.

IX.5.2 *Fonctionnalités*

Nous verrons, dans cette section, les fonctionnalités offertes par notre prototype. Nous pouvons les diviser en deux grandes catégories, les fonctionnalités offertes aux utilisateurs et les fonctionnalités offertes à l'administrateur.

IX.5.2.1 Fonctionnalités utilisateur

Les fonctionnalités offertes aux utilisateurs concernent l'exploration des connaissances contenues dans les ontologies définies et la récupération des données associées. L'ensemble des fonctionnalités est illustré dans la Figure 45 représentant un diagramme de cas d'utilisation.

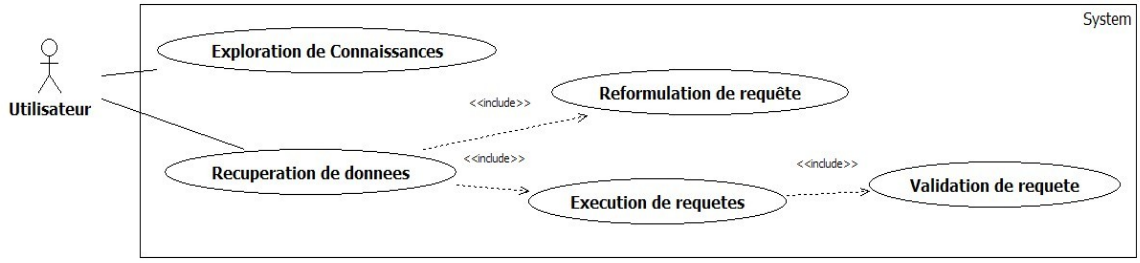

Figure 45: Fonctionnalités utilisateur.

IX.5.2.1.1 *Exploration des connaissances et récupération des données*

L'exploration des connaissances se fait par le lancement d'un client, que nous avons implémenté sous forme d'un agent. Cet agent utilisateur, lors de son lancement récupére la

liste des ontologies disponibles sur le serveur (l'*agent ontologies*). L'utilisateur peut ensuite demander à explorer une ontologie en particulier. L'agent utilisateur peut demander à l'*agent ontologies* de lui fournir la structure de l'ontologie demandée. Cette structure est fournie sous forme d'un objet représentant un document XML. Ce document est parsé et le résultat est présenté à l'utilisateur sous forme d'un graphe. Les noeuds du graphe représentent les concepts de l'ontologie. Les relations sont représentées par des arêtes entre les différents noeuds. Les propriétés sont indiquées sous la forme d'une arborescence. La Figure 46 est une capture d'écran du prototype qui montre le graphe obtenu pour l'ontologie décrivant la maintenance des équipements.

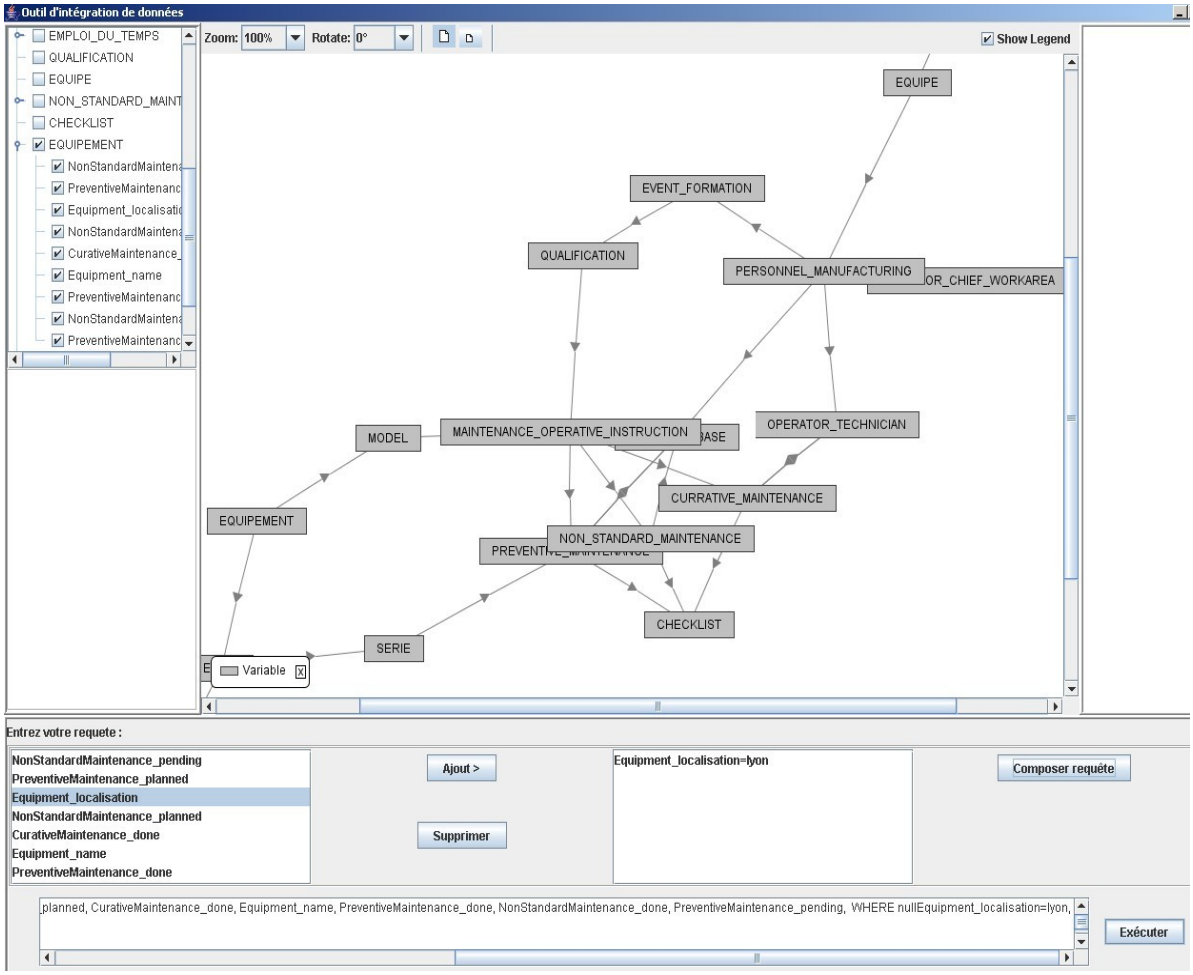

Figure 46: Interface d'exploration des connaissances.

À partir de cette même interface, l'utilisateur peut formuler des requêtes en sélectionnant les

concepts ou propriétés à partir de l'arborescence. Il peut aussi spécifier pour chaque propriété sélectionnée des contraintes. Le protocole d'interaction correspondant au processus de récupération de données est illustré dans la Figure 47 suivante.

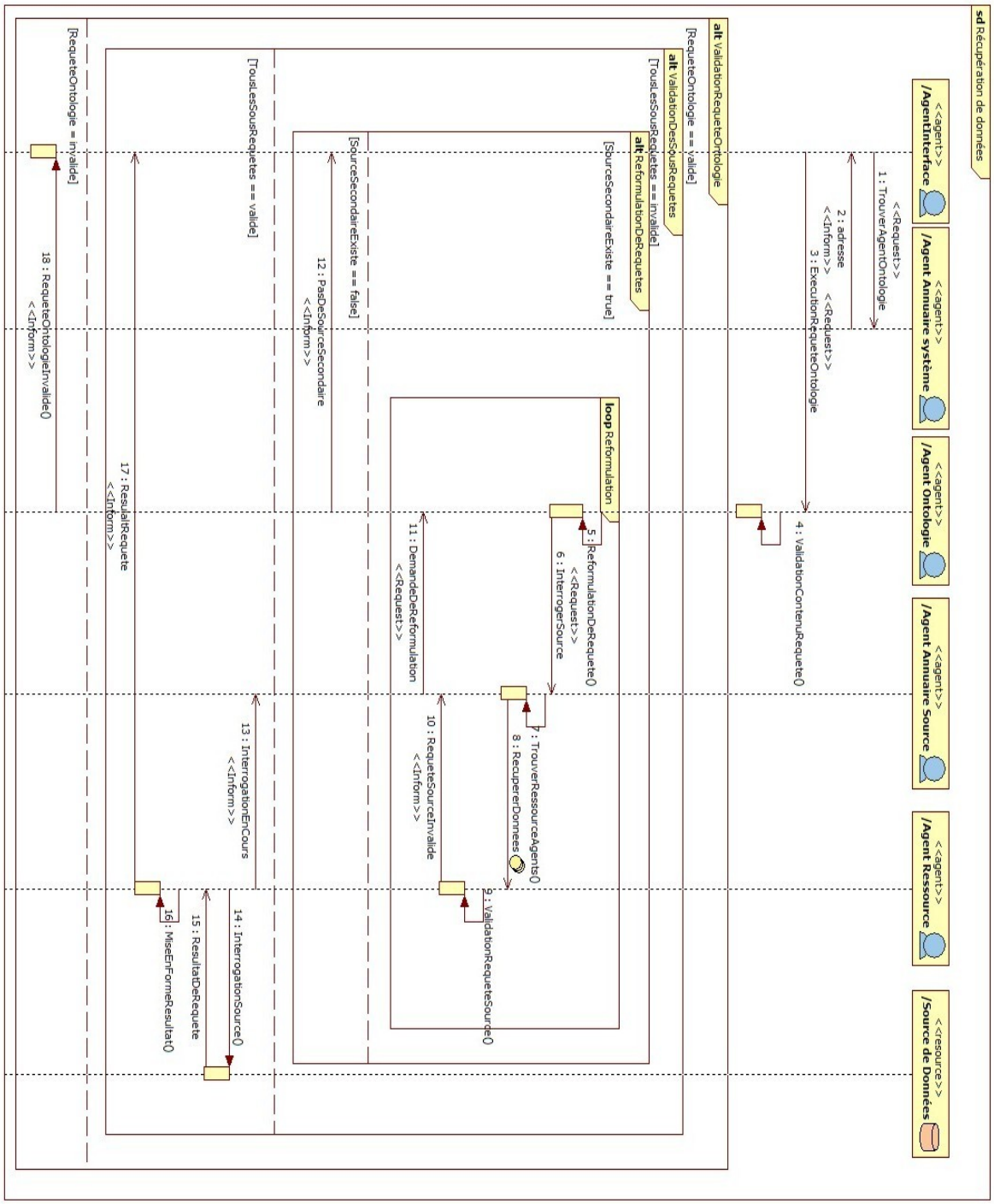

Figure 47: Protocole de récupération des données.

Afin d'obtenir les données, la requête est envoyée à l'*agent ontologies* qu'il aura préalablement

localisé (1:TrouverAgentOntologie). L'agent utilisateur enverra un message avec la requête sous forme d'une chaîne de caractères à l'*agent ontologies* (3:ExecutionRequeteOntologie). Ce dernier vérifie (4:ValidationContenuRequete) que les éléments de la requête font toujours partie de l'ontologie et la reformule (5:ReformulationRequete) en un ensemble de requêtes. L'*agent ontologies* localisera ensuite les agents ressources pertinents (7:TrouverAgentRessource) et leur enverra un ordre d'exécution de requête (8:RecupererDonnees). Le message envoyé aux agents ressources prend la forme d'un objet représentant un document XML de la forme suivante.

```
<?xml version="1.0" encoding="UTF-8"?>
<queryRequest>
  <query>select nom, cp, rue, ville, salaire_net, prenom from salarie,
fiche_de_paie where nom="ghurbhurn" and
fiche_de_paie.idfp=salarie.idpaie</query>
  <joinParameters>
    <joinDb>personne</joinDb>
    <joinAttribut>
      <type>simple</type>
      <attribut>
        <name>nom</name>
      </attribut>
    </joinAttribut>
    <joinAttribut>
      <type>concatenation</type>
      <attribut>
        <name>cp</name>
      </attribut>
      <attribut>
        <name>rue</name>
      </attribut>
      <attribut>
        <name>ville</name>
      </attribut>
    </joinAttribut>
    <joinAttribut>
      <type>simple</type>
      <attribut>
        <name>prenom</name>
      </attribut>
    </joinAttribut>
  </joinParameters>
</queryRequest>
```

Figure 48: Message d'exécution de récupération de données

L'objet contient la requête reformulée ainsi que les attributs (joinAttribut) nécessaires pour la fusion des différentes sous requêtes. Ces « joinAttribut » sont issus des relations intersources (cf Définition VI.4.B page 74) définies par l'administrateur lors de la conception de la représentation métier. Les éléments de la requête sont parsés et les agents ressources valident

les éléments de chaque partie de la requête. En cas d'invalidité l'*agent ontologies* est notifié; dans le cas contraire les requêtes sont exécutées (9:ValidationRequeteSource à 15:ResultatRequete). Le résultat obtenu est formaté sous forme d'un objet décrivant un document XML contenant les données. Un exemple de fichier résultat est donné dans la figure ci-dessous.

```
<?xml version="1.0" encoding="UTF-8"?>
<result>
  <agent>ressourceAg2</agent>
  <database>db</database>
  <requete>select nom, cp, rue, ville, salaire_net, prenom from salarie, fiche_de_paie where
nom="ghurbhurn" and fiche_de_paie.idfp=salarie.idpaie</requete>
  <tuple>
    <joinAttribut>
      <type>simple</type>
      <attribut>
        <name>nom</name>
        <value>ghurbhurn</value>
      </attribut>
    </joinAttribut>
    <joinAttribut>
      <type>concatenation</type>
      <attribut>
        <name>cp</name>
        <value>69002</value>
      </attribut>
      <attribut>
        <name>rue</name>
        <value>rue bichat</value>
      </attribut>
      <attribut>
        <name>ville</name>
        <value>Lyon</value>
      </attribut>
    </joinAttribut>
    <joinAttribut>
      <type>simple</type>
      <attribut>
        <name>prenom</name>
        <value>rahee</value>
      </attribut>
    </joinAttribut>
    <tupleData>
      <salaire_net>1300</salaire_net>
    </tupleData>
  </tuple>
</result>
```

Figure 49: Structuration des données récupérées

Ces objets sont renvoyés à l'agent annuaire source de données pour la fusion qui le renvoie à l'utilisateur (16:MiseEnFormeDonnées). La fusion des documents se fait par une requête

XQUERY sur les « joinAttribut » définis dans chaque document. Le résultat final est aussi renvoyé sous forme d'un objet décrivant les données en XML. La Figure 47 illustre le protocole suivi pour récupérer les données à partir d'une ontologie.

IX.5.2.2 Fonctionnalités administrateur

Les fonctionnalités administrateur sont celles qui lui permettent de construire une ontologie et de la maintenir. En plus de ces fonctions, le prototype lui permet aussi de connaître les changements qui ont été effectués au niveau des sources de données et si une intervention est nécessaire pour la mise à jour de l'ontologie. Les fonctionnalités administrateur sont décrites dans le use case suivant.

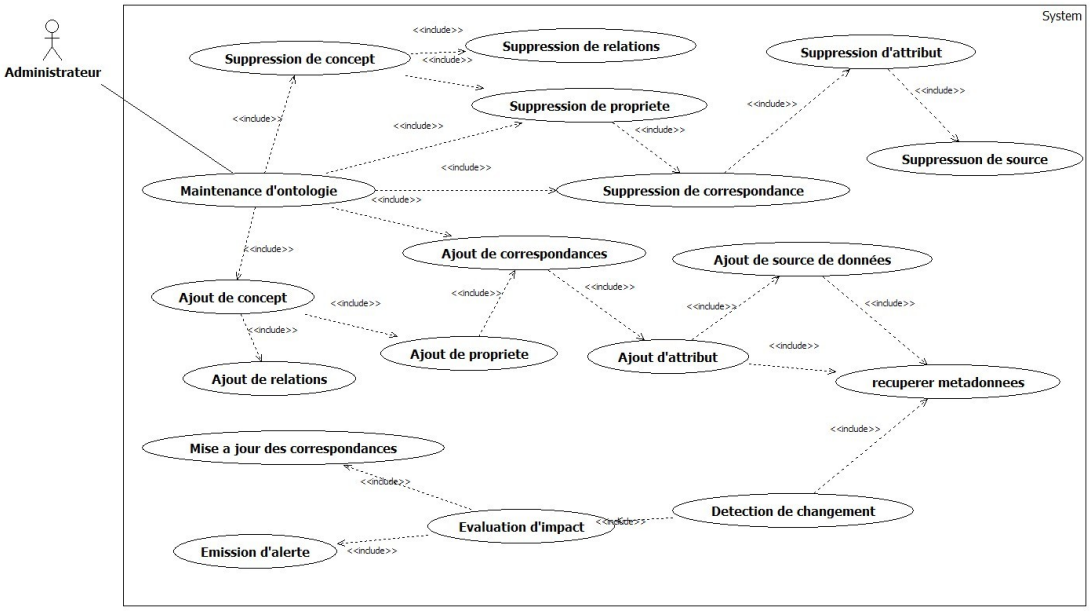

Figure 50: Fonctionnalité administrateur

IX.5.2.2.1 *Maintenance d'ontologie*

Une grande partie des fonctions associées à la maintenance d'ontologie sont remplies par Protégé, un éditeur d'ontologie. En effet, toutes les fonctions de suppression de concepts, de

propriétés, de relations, de correspondances et d'attributs peuvent être accomplies par l'intermédiaire de cet outil. De plus, Protégé permet la création et la modification des éléments cités. *Nous nous concentrons donc sur la récupération des métadonnées et leur ajout au sein de l'ontologie*. Le processus d'ajout de métadonnées peut se faire lors du déploiement d'un *agent ressource* ou en demandant expressément à l'agent de récupérer tout ou partie des métadonnées concernant une source de données.

L'ajout de métadonnées se fait par l'intermédiaire de l'*agent ontologies* qui dispose de l'ontologie à laquelle on souhaite ajouter les métadonnées. *Les agents ressources sont interrogés pour l'obtention des métadonnées. À la réception des informations demandées, l'API Jena est utilisée pour la modification de l'ontologie existante. En effet, nous utilisons l'API, par l'intermédiaire de l'agent ontologies, pour créer des instances des concepts « Attribut » et « Source » si nécessaire*. Ces instances créées, l'ontologie est enregistrée et présentée à l'administrateur pour qu'il puisse définir les correspondances entre les propriétés et les sources de données.

IX.5.2.2.2 *Détection de changements*

La détection de changements est prise en charge entièrement par le système. La seule action qui nécessite l'intervention de l'administrateur est le déploiement d'un agent ressource pour permettre la surveillance de la source de données. Ce processus de déploiement nécessite la spécification des paramètres de connexion à la source de données ainsi que d'un port d'écoute. Comme nous l'avons précisé dans le chapitre VI, nous avons choisi d'émettre un message pour toutes les actions entrainant une modification de structure. Ces messages sont émis par une procédure sur un port donné.

IX.5.2.2.2.1 Déploiement d'agents ressources

Le déploiement d'agents ressources est la première étape pour récolter les métadonnées et détecter les changements. Le protocole d'interaction est illustré dans la Figure 52. L'administrateur commence par créer un agent (1:CreationAgentOntologie) et lui demande de se déployer au niveau d'une source de données en lui fournissant les paramètres de connexion nécessaires. Les paramètres sont spécifiés par l'intermédiaire d'un fichier XML structuré de la

manière suivante.

```
<datasource>
  <type>oracle</type>
  <driver>oracle.jdbc.driver.OracleDriver</driver>
  <host>localhost</host>
  <name>ST</name>
  <login>unLogin</login>
  <password>MotDePasse</password>
  </datasource>
```

Figure 51: Structuration d'un fichier de paramètres

L'*agent ressource* va vérifier s'il existe une source de données à l'url précisée et s'il peut se connecter à la source de données (5:VerificationSource et 6:VerificationAutorisation). Il vérifiera en dernier lieu s'il a les droits suffisants pour utiliser la source de données (9:verificationDeLaSource). Une fois ces vérifications effectuées l'*agent ontologies,* peut déployer le client qui permettra les notifications de changements (12:DeployerClient). Une fois déployé, l'*agent ressource* peut activer le serveur pour la réception des notifications de changements.

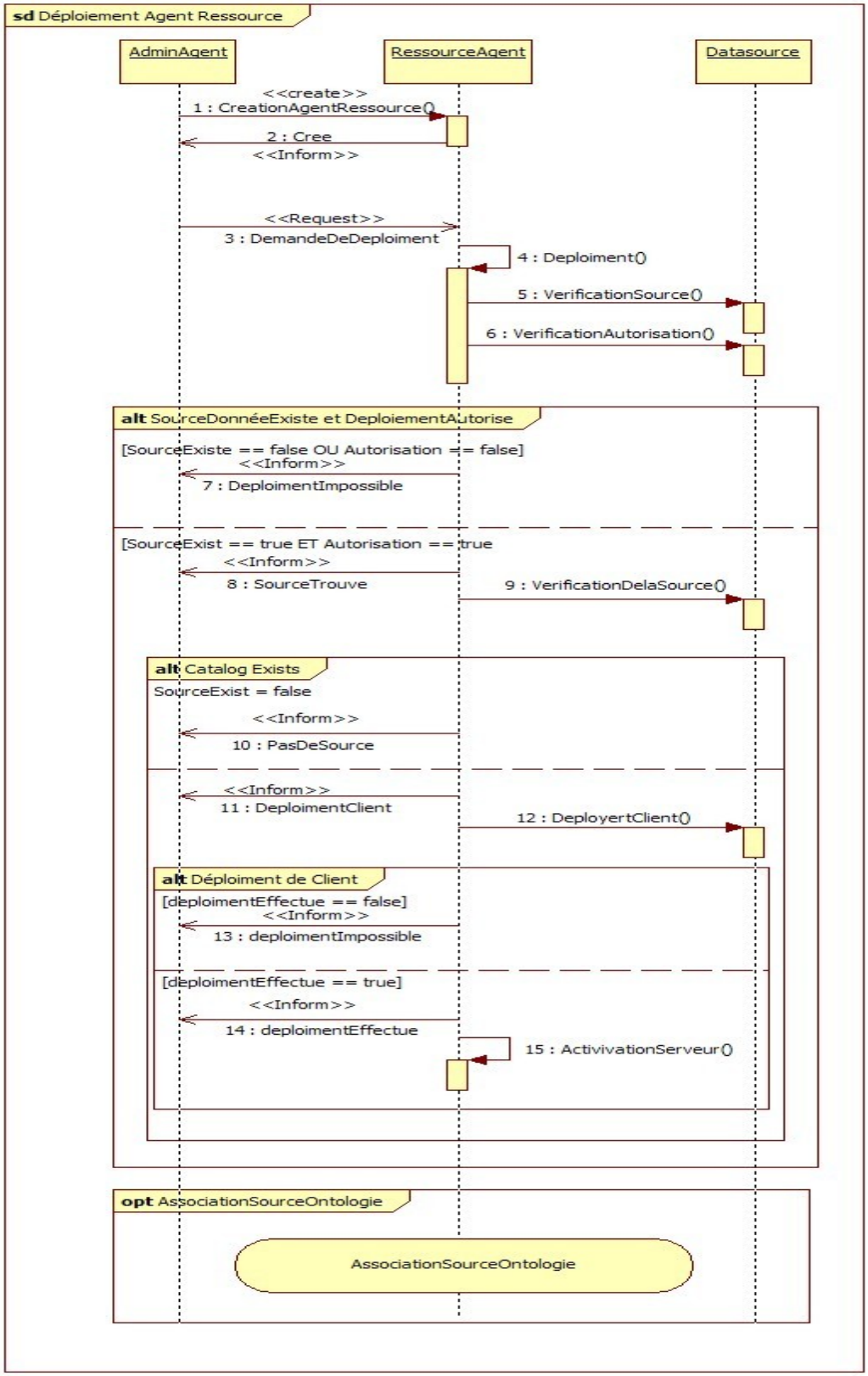

Figure 52: Protocole de déploiement d'un agent ressource

IX.5.2.2.2.2 Mise à jour d'ontologie

Les agents ressources déployés permettent de connaître les modifications effectuées au niveau des sources de données à partir du client déployé au sein de la source de données. Le protocole d'interaction correspondant à la mise à jour d'ontologie est illustré par la Figure 53. Le processus est déclenché par l'émission d'un message par le client indiquant qu'un changement a été effectué (1:ChangementDetecte). L'*agent ressource* informe l'*agent ontologies* de la modification (2:SourceDeDonneesModifiee) et ce dernier évalue l'impact que ce changement a sur l'ontologie (3:EvaluationImpact). Deux cas de figure peuvent se produire soit l'*agent ontologies* est capable de mettre à jour l'ontologie, soit la mise à jour nécessite l'intervention de l'administrateur. Dans le premier cas, l'API Jena est utilisée pour supprimer les correspondances ou les mettre à jour. La mise à jour effectuée, un message est envoyé à l'administrateur. Ce message contient la description du changement effectué au niveau de la source de données et la description de la mise à jour. Dans le deuxième cas, l'administrateur mettra à jour l'ontologie à partir de Protégé.

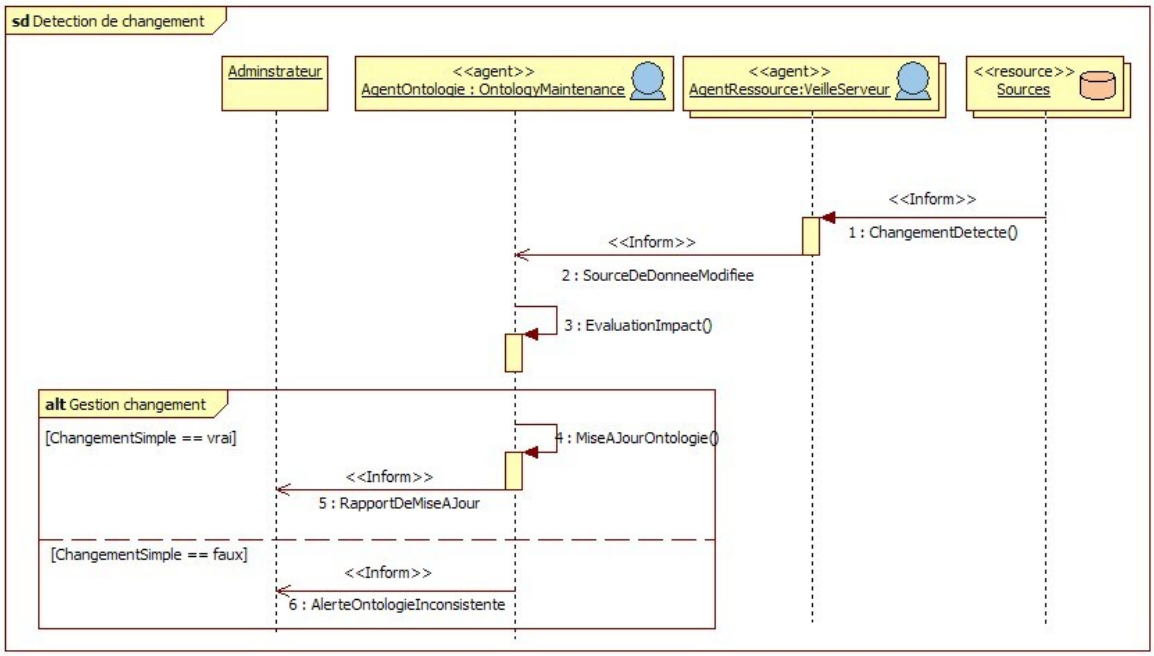

Figure 53: Protocole de détection de changement

IX.6 Conclusion

Nous avons détaillé dans ce chapitre les différents APIs que nous avons utilisée pour la mise en œuvre de notre proposition. L'implémentation du prototype a été fait en JAVA et utilise différentes API pour le déploiement d'agents, la manipulation des ontologies et des documents XML. Nous avons aussi décrit les trois principaux protocoles d'interaction pour la récupération de données, le déploiement d'*agent ressource* et la détection de changement.

Chapitre X.

Conclusion

Nous avons décrit dans cette thèse nos travaux sur l'intégration sémantique de données. Cette problématique à été proposée par STMicroelectronics qui souhaite favoriser la découverte de connaissances présentes au sein de ses systèmes d'informations.

Nous avons commencé par nous positionner dans un contexte général qui est l'évolution des entreprises. En effet, les évolutions du contexte économique d'une entreprise entrainent l'adaptation des stratégies en termes de gamme de produit, implantation sur les marchés et en terme de partenariats. Ces adaptations entraînent des changements en termes de besoins informationnels et en termes de langage utilisé au sein des métiers de l'entreprise.

Les systèmes d'information étant les principaux supports pour la prise de décision, ils sont directement impactés par tous ces changements. Afin de prendre en compte la diversité des sources de données au sein d'une entreprise, un système d'intégration est utilisé pour donner une vision transversale et centralisée des données disséminées. Un tel système est complexe à mettre en œuvre et à maintenir puisqu'il nécessite la prise en compte de l'hétérogénéité des données présentes dans les différents systèmes, mais aussi la surveillance des systèmes qui participent au système d'intégration. Dans cette thèse, nous nous sommes intéressés à la mise en place d'un système d'intégration basé sur la connaissance métier et sur la surveillance des sources de données participant à l'intégration.

Dans la première partie de cette thèse, nous avons passé en revue les différents composants d'un système d'intégration, à savoir un schéma global, des correspondances avec les sources de données, et une architecture.

Le *chapitre II est consacré à l'architecture que peut avoir un système d'intégration*, elle peut

être centralisée suivant une approche médiateur, ou décentralisée suivant une approche PDMS. Quelque soit le type d'architecture, le système d'intégration est soumis à des changements au niveau du schéma global, ou à des évolutions au sein des sources de données du fait des évolutions dans les métiers de l'entreprise. Ces changements n'ont pas d'effets sur l'architecture mais sur les correspondances qui relient le schéma global et les sources de données.

Dans le *chapitre III, nous avons vu les différentes méthodes d'expression d'un schéma global*. Le **modèle relationnel** est très optimisé pour l'interrogation mais *supporte mal l'hétérogénéité* des structures. De plus, il est peu flexible par rapport aux évolutions. Le **modèle objet** offre une meilleure prise en compte de l'hétérogénéité des données mais présente l'inconvénient d'être *peu optimisé pour l'interrogation*. **XML** offre un très bon compromis entre le modèle relationnel et le modèle objet dans la mesure où il permet la prise en compte de l'hétérogénéité des données et permet une interrogation du modèle optimisée. L'inconvénient principale de XML est qu'il *permet de structurer les données mais pas d'expliciter leur sémantique*. Les **logiques de description** permet, quant à elle, une *grande flexibilité en matière de prise en compte de l'hétérogénéité et permet aussi d'expliciter la sémantique des données*. Nous avons donc choisi de modéliser les connaissances métiers en utilisant OWL-DL qui est basé sur une logique de description.

Dans le *chapitre IV, nous avons vu les différents modes d'expression des correspondances* entre un schéma global et des sources de données. Ces correspondances peuvent être de type GAV, LAV, GLAV et BAV. Le **GAV** permet d'exprimer le schéma global en fonction des sources. C'est est une solution *intéressante si les sources de données sont stables*, c'est-à-dire qu'elles ne changent que très rarement. Le **LAV** permet de modéliser les sources en fonction du schéma global. Cette approche n'est pas sensible aux changements au niveau des sources de données mais elle est *sensible à ceux qui se produisent au niveau du schéma global*. De plus, la reformulation des requêtes pour la récupération des données est plus complexe que celle que propose le GAV. Le **GLAV** est une extension du LAV dans laquelle des règles sont ajoutées pour transformer les expressions du LAV en GAV. Le GLAV *hérite des inconvénients du LAV*. Finalement, le **BAV** consiste à définir une séquence de transformation des sources de données pour arriver à un schéma global. Cette approche est intéressante dans la mesure où il est possible de gérer les changements au niveau du schéma global mais aussi au niveau des sources de données. Cependant, *la seule implémentation qui a été proposée est*

coûteuse à mettre en place. En effet, pour prendre en compte les changements au niveau des sources de données et du schéma global, il est nécessaire de recalculer les séquences de transformation. Nous avons choisi de suivre la méthode BAV et proposons une autre façon de modéliser les correspondances qui nous évite de recalculer des transformations.

Le *chapitre V est consacré aux systèmes d'intégration existants*. Nous analysons ces systèmes suivant trois critères, la détection et la gestion du changement, la prise en compte de l'utilisateur et la récupération des données. Aucun système ne permet de détecter les changements au niveau des sources de données. *Seuls deux d'entre eux, Infosleuth et Automed, proposent la gestion du changement mais seul Automed propose la mise à jour des correspondances. En matière de description sémantique du contenu des sources de données, seul MOMIS et SEWASIE proposent un schéma global explicite. Finalement, lors de la récupération des données, seul MOMIS propose une validation syntaxique des données mais aucune proposition ne valide le contenu des requêtes à exécuter.* Nous avons choisi d'avoir au sein de notre proposition, KRISMAS, un mécanisme de détection de changements, un schéma global qui explicite le contenu des sources de données et la validation des requêtes d'un point de vue syntaxique mais aussi au niveau du contenu.

La deuxième partie de cette thèse est consacrée à KRISMAS, notre proposition en termes de modélisation de la connaissance et en termes d'architecture basée sur les agents. Nous avons proposé dans le *chapitre VI la structure d'un schéma global faiblement couplé aux sources de données*. C'est-à-dire que la mise à jour des correspondances entre schéma global et sources de données est moins complexe à réaliser que dans les autres approches. Nous avons aussi proposé un algorithme de reformulation qui nous permet de reformuler les requêtes construites à partir du schéma global en requêtes compréhensibles par les sources de données.

Les chapitres *VII et VIII sont consacrés à l'architecture multi-agent que nous proposons*. Nous décrivons le rôle joué par chaque module et leur structuration interne avant de voir leur fonctionnement global. Le chapitre VIII détail les principaux protocoles que nous avons mis en place pour la détection de changements, la récupération de données et l'ajout/suppression de sources de données, ainsi que l'évolution des connaissances.

Le *chapitre IX traite de l'implémentation de nos propositions*. Nous détaillons les APIs que nous avons utilisées. Nous détaillons aussi le processus complet suivi pour chaque protocole défini dans le chapitre VIII.

X.1 Apports, limites et perspectives

Dans la section précédente, nous avons résumé les différents chapitres de cette thèse et nous verrons dans celui-ci un résumé de nos apports. Nous aborderons aussi les limites de notre proposition avant de donner les pistes d'amélioration et de développement du système.

X.1.1 *Apports*

Nos apports se situent à quatre niveaux :

- la modélisation des connaissances,
- la définition des correspondances,
- la gestion du changement et
- de la récupération de données.

Du point de vue de la modélisation des connaissances, nous proposons une exposition des données contenues dans les sources de données en fonction des connaissances métier. *Notre proposition diffère des propositions existantes car elle ne se contente pas d'établir une taxonomie ou d'exposer les données telles qu'elles sont présentes dans les sources de données*. Nous proposons une ontologie qui est construite avec la participation des acteurs métiers. Cette construction d'ontologie précède l'identification des sources de données pertinentes contrairement aux autre approches où les sources de données sont identifiées et le contenu exposé.

En matière de définition de correspondances, notre proposition est plus flexible que celles existantes. En effet, *les correspondances ne sont pas décrites sous forme de vue. Nous nous contentons de spécifier les attributs représentatifs au travers de « liens ».* Ces liens sont composés d'un nom de sources de données, d'une table (dans le cas de sources de données relationnelles) ou d'un élément (dans le cas de sources de données XML) et d'un nom d'attribut/élément. *L'apport de réponses à des requêtes ne s'apparente donc pas à répondre à des requêtes à partir de vues mais prend la forme de composition de requêtes qui sont exécutées au niveau de chaque source de données. Cette approche nous permet d'introduire la vérification de contenu de requêtes. Cette vérification est un des mécanismes de détection de changement au niveau des sources de données.*

La détection de changements est selon nous un aspect important d'un système d'intégration. Si

les changements ne sont pas gérés, il peut y avoir des incohérences entre le schéma global et les sources de données. Dans le paragraphe précédant nous avons déjà mentionné un des mécanismes de détection de changement à travers la vérification de contenu des requêtes. *Le deuxième mécanisme que nous introduisons correspond aux agents ressources qui réceptionnent les notifications de changements émises par les sources de données. Ce processus déclenche automatiquement une évaluation d'impact au niveau du schéma global et éventuellement une mise à jour ou une alerte.*

Finalement, en termes de récupération de données, *nous introduisons la possibilité d'interroger des sources de données secondaires en cas d'indisponibilité des sources de données principales*. Ce mécanisme est aussi utilisé dans l'éventualité où les données ne peuvent plus être récupérées du fait de modifications de structure au sein des sources de données.

X.1.2 *Limites*

Nous verrons dans cette section les différentes limites de notre proposition au niveau des sources de données.

Au niveau des sources de données, il existe deux limitations :

- le type de sources de données pris en compte ;
- la gestion des l'hétérogénéité des types de données.

A l'heure actuelle, notre proposition permet l'intégration de données à partir de sources de données relationnelles. *La prise en compte d'autre types de sources de données telles que les documents XML ou les fichiers plat posent certains problèmes notamment en matière de détection de changements*. En effet, le mécanisme que nous proposons repose sur l'utilisation des déclencheurs. Or il n'existe pas un tel mécanisme au niveau des documents XML ou des fichiers plats. Il est donc impossible d'émettre des alertes vers les agents ressources en cas de modifications de structures. *Cependant l'utilisation du deuxième mécanisme, la détection par validation du contenu des requêtes, reste toujours possible.*

La deuxième limitation au niveau des sources de données se situe au niveau de la gestion de l'hétérogénéité des types de données. Pour les besoins d'implémentation, *nous avons fait l'hypothèse qu'il existait des attributs dans les différentes sources de données ayant le même type et permettant de relier les sources entre elles*. Cependant, cette hypothèse peut s'avérer

injustifiée entrainant un besoin de transformation de type de données afin de les rendre compatibles en eux. De plus, les données renvoyées aux utilisateurs ont le même type que celui de la source de données où elles ont été extraites. Or l'utilisateur peut vouloir disposer de ces données avec des types différents.

Au niveau de l'architecture que nous proposons, une limite serait sa nature centralisée. En effet, l'agent ontologie est responsable d'une grande partie des traitements. Mais d'un point du point de vue de la veille des sources de données nous pouvons dire qu'elle est distribuée.

X.1.3 *Perspectives*

Notre proposition peut être enrichie suivant deux axe ; l'enrichissement de l'ontologie métier grâce à des fonctions de conversion et extension vers une architecture PDMS.

Fonctions de conversion : comme nous l'avons explicité dans la section précédente, une des limites de notre proposition est la gestion de l'hétérogénéité des types au sein des sources de données. *Afin de pallier cette limite, nous envisageons de définir un des fonctions de transformation*. Ces fonctions pourraient être automatiquement appliquées lors de la récupération de données. Nous pouvons aussi envisager de permettre aux utilisateurs de définir des fonctions de transformation avancées permettant de formater les données avant leur restitution.

Extension au PDMS : Telle que nous l'avons défini, KRISMAS est appliqué au sein d'une organisation pour permettre l'intégration de plusieurs sources de données. Nous pouvons envisager l'intégration de sources de données de deux organisations différentes. En effet, dans le cadre de la coopération inter-entreprises, KRISMAS peut être adapté pour automatiser les échanges inter-organisations. Par exemple une organisation peut émettre une commande avec des spécifications issues de son système d'information vers la deuxième organisation. Ce dernier ayant un système d'information structuré différemment, un mécanisme de correspondance pourrait être établi par l'intermédiaire d'une ontologie d'échange. Cette ontologie d'échange décrirait les concepts de l'échange et des correspondances seraient définies vers les systèmes d'informations des deux organisations. Il serait alors possible de traiter automatiquement les flux de commandes entre les deux organisations.

Bibliographie

[1] Julie Chapron, **L'urbanisme organisationnel : méthode et aides à la décision pour piloter l'évolution du système d'information de l'entreprise.**, Département Génie Industriel et Informatique, Ecole Nationale Supérieure des Mines de Saint Etienne, 2006.

[2] Domenico Lembo and Maurizio Lenzerini and Riccardo Rosati, **Review on Models and Systems for Information Integration**, Deliverable D1.1, INFOMIX Consortium, 2002.

[3] Diego Calvanese and Giuseppe De Giacomo and Maurizio Lenzerini, **A Framework for Ontology Integration**, Proc. of the 1st Semantic Web Working Symposium at the Emerging Semantic Web, pp. 201-214, 2002.

[4] Gio Wiederhold, **Mediators in the Architecture of Future Information Systems**,Computer, 38-49, 1992

[5] Gio Wiederhold, **Mediation in information systems**, ACM Comput. Surv.265-267, 1995

[6] Ulf Leser, **Query Planning in Mediator Based Information Systems**, Phd thesis, Informatik der Technischen Universität Berlin, 2000.

[7] Gio Wiederhold and Michael Genesereth, **The Conceptual Basis for Mediation Services**, IEEE Expert: Intelligent Systems and Their Applications, 38-47, 1997

[8] Diego Calvanese and Giuseppe De Giacomo and Maurizio Lenzerini and Riccardo Rosati, **Logical foundations of peer-to-peer data integration**, PODS '04: Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems , 241-251, 2004.

[9] Blanco, R. and Ahmed, N. and Hadaller, D. and Sung, L. G. A. and Li, H. and Soliman, M. A. , **A Survey of Data Management in Peer-to-Peer Systems**, Technical Report CS-2006-18, 1-51,2006

[10] Wee Siong Ng and Beng Chin Ooi and Kian-Lee Tan and Aoying Zhou, **PeerDB: A P2P-based System for Distributed Data Sharing**, ICDE , 633-644, 2003.

[11] Chen-Chuan K. Chang and Hector Garcia-Molina, **Mind your vocabulary: query mapping across heterogeneous information sources**, SIGMOD '99: Proceedings of the 1999 ACM SIGMOD international conference on Management of data , 335-346, 1999.

[12] Zachary G. Ives and Alon Y. Halevy and Peter Mork and Igor Tatarinov, **Piazza: mediation and integration infrastructure for Semantic Web data**, Journal of Web Semantics, vol 1, 155-175, 2004.

[13] Beng Chin Ooi and Yanfeng Shu and Kian-Lee Tan, **Relational data sharing in peer-**

based data management systems, SIGMOD Rec.,59-64, 2003

[14] Craig S. Mullins, **Managing Database Change**, Web Article, http://www.craigsmullins.Com/dbt_1000.htm, 2000.

[15] John F. Roddick and Lina Al-Jadir and Leopoldo Bertossi and Marlon Dumas and Florida Estrella and Heidi Gregersen and Kathleen Hornsby and Jens Lufter and Federica Mandreoli and Tomi Mannisto; and Enric Mayol and Lex Wedemeijer, **Evolution and change in data management : issues and directions**, SIGMOD Rec, 21-25, 2000

[16] Elke A. Rundensteiner and Amy J. Lee and Anisoara Nica, **On Preserving Views in Evolving Environments**, KRDB , 13.1-13.11, 1997.

[17] Peter Pin-Shan Chen, **The entity-relationship model toward a unified view of data**, ACM Trans. Database Syst., 1, 9-36, 1976.

[18] Ramez A. Elmasri and Shamkant B. Navathe, **Fundamentals of Database** Systems, Addison-Wesley Longman Publishing Co., Inc., 1998.

[19] E. F. Codd, **A relational model of data for large shared data banks**, Commun. ACM64-69, 1983

[20] Raghu Ramakrishnan and Johannes Gehrke, **Database Management Systems**, McGraw-Hill Science/Engineering/Math, 2002.

[21] Jeffrey D. Ullman, **Principles of Database Systems**, W. H. Freeman & Co., 1983.

[22] E. F. Codd, Relational Completeness of Data Base Sublanguages.,In: R. Rustin (ed.): **Database Systems**: 65-98, Prentice Hall and IBM Research Report RJ 987, San Jose, California 1972

[23] D. Beneventano and S. Bergamaschi and F. Mandreoli and M. Marchica, **Query Manager functional specifications**, Deliverable D1.R8, D2I Project, www.dis.uniroma1.it/~lembo/D2I/Prodotti/deliverable2/D1.R8.ps , 2001.

[24] Yannis Papakonstantinou and Hector Garcia-Molina and Jennifer Widom, **Object Exchange Across Heterogeneous Information Sources**, ICDE '95: Proceedings of the Eleventh International Conference on Data Engineering , 251-260, 1995.

[25] Sonia Bergamaschi and Silvana Castano and Maurizio Vincini and Domenico Beneventano, **Semantic integration of heterogeneous information sources**, Data Knowl. Eng.215-249, 2001

[26] R. G. G. Cattell and Douglas K. Barry, **The Object Data Standard: ODMG 3.0**, Morgan Kaufmann, 2000.

[27] Domenico Beneventano and Sonia Bergamaschi and Stefano Lodi and Claudio Sartori,

Consistency Checking in Complex Object Database Schemata with Integrity Constraints, IEEE Transactions on Knowledge and Data Engineering576-598, 1998

[28] S. Bergamaschi and B. Nebel, **Acquisition and Validation of Complex Object Database Schemata Supporting Multiple Inheritance**, Applied Intelligence, 4, 185-203, 1994.

[29] Antoine Galland, **Représentation de schéma pour données semi-structurées**, Mémoire de DEA, INRIA, 2000.

[30] Tim Furche and François Bry and James Bailey and Sebastian Schaffert and Benedikt Linse and Renzo Orsini and Ian Horrocks and Michael Kraus and Oliver Bolzer, **Survey over Existing Query and Transformation Languages - Revision 2.0**, Deliverable I4-D1a, http://rewerse.net/deliverables/m24/i4-d9a.pdf, 2006.

[31] Bertram Ludascher and Yannis Papakonstantinou and Pavel Velikhov, **A Brief Introduction to XMAS**, http://www.npaci.edu/DICE/Pubs/XMAS-intro.pdf, 1999.

[32] M. Ross Quillian, **Semantic Memory**, Carnegie Institute of Technology, Pittsburgh, PA, 1966.

[33] Marvin Minsky, **A Framework for Representing Knowledge**, In P. Winston, editor, The Psychology of Computer Vision, pages 211–277. McGraw-Hill, New York, 1975.

[34] Ronald J. Brachman, Structured inheritance networks1978

[35] Alexander Borgida and Ronald J. Brachman and Deborah L. McGuinness and Lori Alperin Resnick, **CLASSIC: a structural data model for objects**, SIGMOD '89: Proceedings of the 1989 ACM SIGMOD international conference on Management of data , 58-67, 1989.

[36] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, Peter Patel-Schneider, **The description logic handbook: theory, implementation, and applications**, Cambridge University Press, 2003.

[37] Robert MacGregor and Raymond Bates, **The Loom Knowledge Representation Language**, Technical Report, ISI/RS-87-188, USC/Information Sciences Institute, Marina del Rey, Calif. 1987.

[38] Robert M. MacGregor, **A description classifier for the predicate calculus**, AAAI '94: Proceedings of the twelfth national conference on Artificial intelligence (vol. 1) , 213-220, 1994.

[39] Robert M. MacGregor, **Inside the LOOM description classifier**,SIGART Bull., 2, 88-92, 1991.

[40] Alon Y. Levy and Marie-Christine Rousset, **CARIN: A Representation Language Combining Horn Rules and Description Logics**. ECAI , 323-327, 1996.

[41] Vinay K. Chaudhri and Adam Farquhar and Richard Fikes and Peter D. Karp and James Rice, **OKBC: A Programmatic Foundation for Knowledge Base Interoperability**. AAAI/IAAI , 600-607, 1998.

[42] Eric Mays and Robert Dionne and Robert Weida, **K-Rep system overview**,SIGART Bull., 2, 93-97, 1991.

[43] Christof Peltason, **The BACK system : an overview**, SIGART Bull,114-119, 1991

[44] Bernhard Nebel, **Reasoning and revision in hybrid representation systems**, Springer-Verlag New York, Inc., 1990.

[45] Bernhard Nebel, **Terminological Reasoning is Inherently Intractable**, Articial Intelligence Journal, 43, 235-249, 1990.

[46] Peter F. and Patel-Schneider, **DLP System Description.**, Description Logics, 1998.

[47] Ian Horrocks, **FaCT and iFaCT.**, International Workshop on Description Logics, 1999.

[48] V. Haarslev and R. Moller, **RACE System Description**, Proc. of DL99, International Workshop on Description Logics, 130-132, 1999.

[49] Ora Lassila, **Web Metadata: A Matter of Semantics**, IEEE Internet Computing, 2, 30-37, 1998.

[50] Ora Lassila and Ralph Swick, **Resource Description Framework (RDF) Model and Syntax Specification**, http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/ , 1999.

[51] D. Brickley and R.V. Guha, **Resource Description Framework (RDF) Schema Specification 1.0**, http://www.w3.org/TR/rdf-schema/, 2004.

[52] J. Hendler and D. L. McGuinness, **The DARPA Agent Markup Language**, IEEE Intelligent Systems, 67-73, 2000

[53] Sean Bechhofer and Jeen Broekstra and Stefan Decker and Michael Erdmann and Dieter Fensel and Carole Goble and Frank van Harmelen and Ian Horrocks and Michel Klein and Deborah McGuinness and Enrico Motta and Peter Patel-Schneider and Steffen Staab, and Rudi Studer, **An informal description of Standard Oil and Instance OIL**, White paper, www.ontoknowledge.org/oil/downl/oil-whitepaper.pdf , 2000.

[54] Deborah L. Mcguinness and Frank van Harmelen, **OWL Web Ontology Language Overview**, http://www.w3.org/TR/owl-features/ , 2004.

[55] Ian Horrocks and Peter F. Patel-Schneider, **Reducing OWL Entailment to Description Logic Satisfiability**. International Semantic Web Conference , 17-29, 2003.

[56] Mohand-Said Hacid and Chantal Reynaud, **L'intégration de sources de données**, Revue Information - Interaction – Intelligence (I3) Une Revue en Sciences du Traitement de l'Information, 2005

[57] Zohra Bellahsene, **Schema evolution in data warehouses**,Knowledge and Information Systems, 4, 283-304, 2002.

[58] W H Inmon, **Building the Data Warehouse**, John Wiley & Sons, Inc., New York, NY, USA, 2002.

[59] R Kimball and L Reeves and M Ross and W Thornthwaite, **Concevoir et déployer un data warehouse**, Eyrolles, 2000.

[60] Panos Vassiliadis and Alkis Simitsis and Spiros Skiadopoulos, **Conceptual modeling for ETL processes**, DOLAP '02: Proceedings of the 5th ACM international workshop on Data Warehousing and OLAP, 14-21,2002.

[61] Panos Vassiliadis and Alkis Simitsis and Panos Georgantas and Manolis Terrovitis and Spiros Skiadopoulos, **A generic and customizable framework for the design of ETL scenarios**, Inf. Syst., 492-525,2005.

[62] Alon Y. Levy, **Logic-based techniques in data integration**, Kluwer Academic Publishers, 2000.

[63] Ioanna Koffina and Giorgos Serfiotis and Vassilis Christophides, F**oundations for Information Integration : A State of the Art**, Technical report, http://delos.di.uoa.gr/transactions.php?type=Reports,2005.

[64] Jeffrey D. Ullman, **Information Integration Using Logical Views**, ICDT '97: Proceedings of the 6th International Conference on Database Theory , 19-40, 1997.

[65] Maurizio Lenzerini, **Data integration: a theoretical perspective**, PODS '02: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems , 233-246, 2002.

[66] Marc Friedman and Alon Levy and Todd Millstein, **Navigational plans for data integration**, AAAI '99/IAAI '99: Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence , 67-73, 1999.

[67] Diego Calvanese and Domenico Lembo and Maurizio Lenzerini, **Survey on methods for query rewriting and query answering using views**, Tecnical report, University of Rome 'La Sapienza', 2001.

[68] Héctor Manuel Pérez-Urbina, **Logic-based information integration**, Report, http://www.cs.man.ac.uk/~perezurh/files/long.pdf, 2007.

[69] A. Cali and G. De Giacomo and M. Lenzerini, **Models for information integration: Turning local-as-view into global-as-view**,Proc. of Int. Workshop on Foundations of Models for Information Integration (10th Workshop in the series Foundations of Models and Languages for Data and Objects), 2001.

[70] Michael Boyd and Sasivimol Kittivoravitkul and Charalambos Lazanitis and Peter McBrien and Nikos Rizopoulos, **AutoMed: A BAV Data Integration System for Heterogeneous Data Sources**., Advanced Information Systems Engineering , 82-97, 2004.

[71] Peter McBrien and Alexandra Poulovassilis, **Data Integration by Bi-Directional Schema Transformation Rules**., Proceedings. 19th International Conference on Data Engineering, 227 – 238, 2003.

[72] Hao Fan and Alexandra Poulovassilis, **Schema Evolution in Data Warehousing Environments - A Schema Transformation-Based Approach**.,ER, 639-653,2004

[73] Marian H. Nodine and Jerry Fowler and Brad Perry, **Active Information Gathering in InfoSleuth**., Cooperative Database Systems and Applications , 15-26, 1999.

[74] Brad Perry and Malcolm Taylor and Amy Unruh, **Information Aggregation and Agent Interaction Patterns in InfoSleuth**, COOPIS '99: Proceedings of the Fourth IECIS International Conference on Cooperative Information Systems , 314, 1999.

[75] Peter McBrien and Alexandra Poulovassilis, **Defining Peer-to-Peer Data Integration Using Both as View Rules**, Databases, Information Systems, and Peer-to-Peer Computing, 91-107, 2003.

[76] Sonia Bergamaschi and Pablo R. Fillottrani and Gionata Gelati, **The SEWASIE Multi-agent System.**, AP2PC , 120-131, 2004.

[77] Paolo Dongilli and Pablo R. Fillottrani and Enrico Franconi and Sergio Tessaris, **A Multi-Agent System for Querying Heterogeneous Data Sources with Ontologies.**,SEBD , 75-86, 2005.

[78] Domenico Beneventano and Sonia Bergamaschi and Francesco Guerra and Maurizio Vincini, **Building an integrated Ontology within SEWASIE system**., SWDB , 91-107, 2003.

[79] Philippe Adjiman, **Peer-to-peer reasoning in propositional logic : algorithms, scalability study and applications**, Mémoire de thèse, Université Paris Sud , 2006.

[80] Marie-Christine Rousset, **Small Can Be Beautiful in the Semantic Web**.,International Semantic Web Conference , 6-16, 2004.

[81] Marie-Christine Rousset and P. Adjiman and Philippe Chatalic and François Goasdoué and Laurent Simon, **SomeWhere in the Semantic Web**., SOFSEM , 84-99, 2006.

[82] Sudarshan S. Chawathe and Hector Garcia-Molina and Joachim Hammer and Kelly Ireland and Yannis Papakonstantinou and Jeffrey D. Ullman and Jennifer Widom, **The TSIMMIS Project: Integration of Heterogeneous Information Sources**., PSJ , 7-18, 1994.

[83] Yigal Arens and Craig A. Knoblock and Wei-Min Shen, **Query Reformulation for Dynamic Information Integration.**, Journal of Intelligent Information Systems, 99-130, 1996

[84] Yigal Arens and Chin Y. Chee and Chun-Nan Hsu and Craig A. Knoblock, **Retrieving and Integrating Data from Multiple Information Sources.**, International Journal on Cooperative Information System,127-158, 1993

[85] Eduardo Mena and Arantza Illarramendi and Vipul Kashyap and Amit P. Sheth, **OBSERVER: An Approach for Query Processing in Global Information Systems Based on Interoperation Across Pre-Existing Ontologies**., Distributed and Parallel Databases223-271, 2000

[86] Domenico Beneventano and Sonia Bergamaschi and Francesco Guerra and Maurizio Vincini, **Synthesizing an Integrated Ontology**, IEEE Internet Computing42-51, 2003

[87] Domenico Beneventano and Sonia Bergamaschi, **The MOMIS methodology for integrating heterogeneous data sources**., IFIP Congress Topical Sessions , 19-24, 2004.

[88] Marie-Christine Rousset and Alain Bidault and Christine Froidevaux and Hélène Gagliardi and François Goasdoué and Chantal Reynaud and Brigitte Safar, **Construction de médiateurs pour intégrer des sources d'information multiples et hétérogènes : le projet PICSEL**, Revue I3, Vol. 2, 9-59, 2002.

[89] Michel Page, Jérôme Gensel, Cécile Capponi, Christophe Bruley, Philippe Genoud, Danielle Ziébelin, **Representation de connaissances au moyen de classes et d'associations : le systeme AROM**LMO, 91-106,2000.

[90] Radovan Cervenka and Ivan Trencansk'y and Monique Calisti and Dominic A. P. Greenwood, **AML: Agent Modeling Language Toward Industry-Grade Agent-Based Modeling**AOSE, 31-46,2004.

[91] Jeremy J. Carroll and Ian Dickinson and Chris Dollin and Dave Reynolds and Andy Seaborne and Kevin Wilkinson, **Jena: implementing the semantic web recommendations**WWW Alt. '04: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters, 74-83,2004.

[92] P. D. O'Brien and R. C. Nicol, **FIPA : Towards a Standard for Software Agents**, BT Technology Journal51-59, 1998

Ecole Nationale Supérieure des Mines
de Saint-Etienne

N° d'ordre : 480 I

Rahee Ghurbhurn

Knowledge based data integration – a multiagent approach for handling changes

PhD in Computer Science

Keywords : data integration, change management, multiagent system, ontology, information retrieval

Abstract :

Organisations are generally composed of several information systems which are key components in the decision making process. Finding specific functional information in these systems may prove to be difficult for non IT users. Data integration systems are used to centralise data from several information systems but the actual solutions simply expose the collected data and don't manage changes that may occur in the underlying data sources structure. The objective of our work is to express the collected data in terms of business knowledge and allow functional users to explore knowledge and retrieve the corresponding data without having to know the underlying data sources structures. These data are found in distributed and heterogeneous data sources belonging to different services or departments. Our proposal, KRISMAS, is based on ontologies modelling functional domains and a multi-agent architecture performing the data retrieval and managing the changes that may occur within the data sources. By managing the changes we mean maintaining the ontologies coherent with the data sources.

Ecole Nationale Supérieure des Mines
de Saint-Etienne

N° d'ordre : 480 I

Rahee Ghurbhurn

Intégration de données à partir de connaissances – une approche multi-agent pour la gestion des changements

Spécialité : informatique

Mots clefs : intégration de données, gestion des changements, système multi-agent, ontologie, web sémantique

Résumé :

Au sein des entreprises, un composant vital dans la prise de décision, qu'elle soit globale ou locale, est le système d'information. Celui-ci contient, en effet, les informations nécessaires pour décider, agir, apprendre, comprendre, prévoir et contrôler. Sa structure est généralement liée à l'histoire de l'entreprise dans le sens où cet ensemble s'est constitué d'éléments qui se sont juxtaposés au fil du temps au gré des choix stratégiques formant in fine un ensemble complexe et hétérogène. L'existence de plusieurs systèmes d'information, au sein de grande sociétés, pouvant rendre la recherche d'information difficile pour les utilisateurs métiers. L'objectif de nos travaux consiste à permettre aux utilisateurs d'explorer les connaissances, formulées en terme métier, contenues dans plusieurs systèmes d'informations et de récupérer les données qui leur sont associées. Il s'agit donc de mettre à disposition des utilisateurs une vue globale des connaissances disponibles liées à leurs domaines métiers, tout en dissimulant la diversité des sources d'informations et en garantissant que les données associées sont récupérables malgré les changements qui peuvent se produire au sein des différents systèmes. Dans cette thèse nous proposons KRISMAS, un solution d'intégration de données basée une représentation des connaissances métiers et une architecture Multi-Agents. La représentation des connaissances prend la forme d'une ou plusieurs ontologies de domaine permettant aux utilisateurs d'explorer les connaissances des sources de données et de formuler des requêtes pour la récupération des données. Les agents sont utilisés pour l'exploitation de la représentation des connaissances métiers ainsi que pour la gestion des changements au sein d'un système d'intégration